Monitoring Environmental Boundaries with a Robotic Sensor Network

Sara Susca   Sonia Martínez   Francesco Bullo
Center for Control, Dynamical Systems and Computation
University of California at Santa Barbara
Santa Barbara, CA, 93106-5070, USA
sara@ece.ucsb.edu,soniamd@ucsd.edu,bullo@engineering.ucsb.edu

Technical report CCDC-06-01-25
Original version: January 25, 2006
Revised version: March 23, 2006

March 23, 2006

Abstract

In this paper we propose and analyze two algorithms to monitor an environmental boundary with mobile sensors. The objective is to optimally approximate the boundary with a polygon. In the first scenario the mobile sensors know the boundary and the approximating polygon is defined by the sensors positions. In the second scenario the mobile sensors rely only on sensed local information to position some interpolation points and define an approximating polygon. For both scenarios we design algorithms that distribute the vertices of the approximating polygon uniformly along the boundary. The notion of uniform placement relies on a metric inspired by known results on approximation of convex bodies. The first algorithm is proved to converge in the case of static boundaries whereas the second one is provably convergent also for slowly-moving boundaries because of certain input-to-state stability properties.

1 Introduction

In recent papers much attention has been given to the problem of boundary estimation and boundary tracking by means of robotic networks. In particular the common goal is to design an algorithm that allows a limited number of mobile sensors to detect the boundary of a region of interest and estimate it as it evolves. The reason of so much attention lies in the numerous applications that boundary estimation and tracking can be used for. Detection of harmful algae bloom [1, 2], oil spill [3], and fire boundary estimation [4, 5] are just some examples. In [1], Bertozzi *et al.* adopt the so called "snake algorithm" from the computer vision literature to detect and track the boundary of harmful algae bloom. The agents are equipped with a chemical sensor that is able to measure concentration gradient and with a communication system that is able to exchange information with a data fusion center. In [2], Kemp *et al.* suggest an algorithm that requires only a concentration sensor: the agents repeatedly cross the boundary using a bang-bang angular velocity controller. In [3] the authors use a random coverage controller, a collision avoidance controller and a bang-bang angular velocity controller to detect and surround an oil spill. In [4, 5] Casbeer *et al.* describe an algorithm that allows LASE (Low Altitude Short Endurance) Unmanned Vehicles to closely monitor the boundary of a fire. The LASEs have an infrared camera and a short range communication device to exchange information with other agents and to download the information collected onto the base station. The algorithm presented will uniformly space the LASEs along the boundary of the fire so that eventually they will patrol an equal portion of the perimeter

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **23 MAR 2006**	2. REPORT TYPE	3. DATES COVERED **00-03-2006 to 00-03-2006**

4. TITLE AND SUBTITLE **Monitoring Environmental Boundaries with a Robotic Sensor Network**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Barbara,Center for Control, Dynamical Systems and Computation,Santa Barbara,CA,93106-5070**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**26**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

of the fire. A different approach is considered by Zhang and Leonard in [6]. A formation of four robots tracks at unitary speed the level sets of a field. Their relative position changes so that they optimally measure the gradient and estimate the curvature of the field in the center of the formation. The center of the formation moves along the level sets.

In this paper we propose two algorithms to estimate and reconstruct the boundary of a region. In the first idealized scenario we assume that the agents have knowledge of the boundary and that they are equipped with GPS and communication devices.

In the second scenario, we require a group of Unmanned Air Vehicles (UAVs) to optimally place some interpolation points on the boundary of a region of interest. The boundary can then be reconstructed by linear interpolation of the interpolation points. We assume that (i) the UAVs do not have a priori knowledge of the boundary, (ii) they are equipped with a camera sensor and with algorithms to estimate the tangent and curvature of the boundary, and (iii) a wireless communication network provides the UAVs with the ability to download and upload the interpolation points from and to a data center. The algorithm is provably convergent also in the case of slowly-moving boundaries because of certain input-to-state stability properties.

The novelty of this paper is in the criterion used to optimally place robots and interpolation points in the first and second scenario in such a way that they are uniformly distributed according to a curvature-weighted distance function defined along the boundary. The curvature-weighted distance function was inspired by the literature on optimal approximation of convex bodies by polygons e.g., see the survey [7, 8].

The convergence of both algorithms is proven using tools from the theory of consensus algorithms. An incomplete list of relevant literature references includes [9], [10], and [11]. In [10], Blondel *et al.* propose an extension to Wolfowitz Lemma used in Jadbabaie *et al.* [9]. The authors prove that an infinite product of stochastic matrices (belonging to an infinite set, and with diagonal terms uniformly lower-bounded by a positive constant) converges to a rank-one matrix (despite bounded communication delays) if the corresponding graphs are jointly strictly connected. In [11], Moreau uses set-valued Lyapunov functions and gives necessary and sufficient conditions to reach consensus in both directed and undirected graphs. It is required that a node is connected to all other nodes (less restrictive condition than the one in [10]) across some bounded or possibly unbounded interval (respectively for undirected and directed graphs), and that the dynamic mapping is strictly convex.

The paper is organized as follows. In Section 2 we review some mathematical literature on approximation theory and convex optimization. In Section 3 we introduce the curvature-weighted deployment algorithm for the case of known boundaries. In Section 4 we introduce an algorithm to jointly update an environment boundary and deploy the UAVs uniformly along the boundary estimate. In Section 5 we present our final concluding remarks.

2 Preliminaries

In this section we review some known useful results from approximation and matrix theory.

2.1 Approximation of convex and non-convex curves

A lot of work has been done to characterize the best polygon approximating a strictly convex body, e.g., see the extensive surveys [7, 8]. In the standard literature on convex bodies approximations, the symmetric difference δ^S between two compact, and strictly convex bodies C, $B \in \mathbb{R}^d$ is defined by

$$\delta^S(C, B) = \mu(C \cup B) - \mu(C \cap B),$$

where μ is the Lebesque measure on $\mathbb{R}^d$. If Q is the body to be approximated by an inscribed n-vertices polygon P_n, then $\delta^S(Q, P_n) = \mu(Q) - \mu(P_n)$. For n sufficiently large, McLure and Vitale [12] show that

$$\delta^S(Q, P_n^*) \approx \frac{1}{12n^2}\left(\int_0^{2\pi} \rho(\theta)^{2/3} d\theta\right)^3 = \frac{1}{12n^2}\left(\int_{\partial Q} \kappa(\ell)^{1/3} d\ell\right)^3,$$

where P_n^* is the best approximating polygon with n vertices inscribed in Q, ∂Q is the boundary of Q, ρ and $\kappa = \rho^{-1}$ are the curvature radius and curvature of the boundary, respectively, ℓ is the arc length along ∂Q, and θ is the angular position in a polar variable parametrization of ∂Q. It is not an easy task to construct the best approximating polygon P_n^* for a strictly convex body, but McLure and Vitale in [12] suggest the *method of empirical distributions*. According to this method, the positions θ_i, $i \in \{1, \dots, n\}$, of the n vertices along ∂Q have the property that $D^S(i) = \int_{\theta_i}^{\theta_{i+1}} \rho(\theta)^{2/3} d\theta$ has the same value for every consecutive pair of vertices $(i, i+1)$. Interpolating polygons computed according to the method of empirical distributions converge to P_n^* as $n \to +\infty$.

For smooth non convex bodies with a small number of saddle points, the method of empirical distributions will also yield a nearly optimal distribution as $n \to +\infty$ because of the local convexity of the body. We show how to do this in what follows. Since the curvature radius ρ may be unbounded at some point of a non-convex boundary, the integral $D^S(i)$ may be unbounded for some i. We avoid this problem by considering the following general notion of distance along a boundary. For $\lambda \in [0, 1]$, we define the pseudo-distance D_λ between vertices $(i, i+1)$ by:

$$D_\lambda(i) = \lambda \int_{\ell_i}^{\ell_{i+1}} \kappa(\ell)^{1/3} d\ell + (1-\lambda)(\ell_{i+1} - \ell_i).$$

This definition is inspired by the fact that, for convex bodies, we have $\int_0^{2\pi} \rho(\theta)^\alpha = \int_{\partial Q} \kappa(\ell)^{1-\alpha} d\ell$, see [7]. Introducing the convex combination with arc length, we guarantee that $D_\lambda(i)$ is non-zero whenever the vertices i and $i+1$ do not coincide. In what follows we shall apply a version of the method of empirical distributions in which the positions of any two consecutive vertices of the interpolating polygon are uniformly distributed according to the pseudo-distance D_λ.

We conclude this section by remarking that a metric similar to D_λ is used by Chen *et al.* in [13, 14]. In this paper, interpolation problems are motivated by certain diffusion phenomena in singularly perturbed systems. The authors show that quasi-optimal interpolation errors in the L_∞ or L_1 norm are achieved when the grid is uniform in the $\int \kappa^{1/2} d\ell$ or $\int \kappa^{1/3} d\ell$ metric, respectively.

2.2 Metzler matrices

We begin by introducing some definitions and notations taken from Chapter 5 in [15]. For $A \in \mathbb{R}^{n \times m}$, we let $A \succ (\succeq) 0$ denote that the elements of A are positive (resp. non-negative). For $A \in \mathbb{R}^{n \times n}$, we let $A >$ (resp. $\geq$)0 denote that the matrix A is positive definite (resp. positive semidefinite) and we let $\rho(A)$ be the spectral radius of A.

Definitions 1. *(i) A square matrix is said to be of class Z if all of its off-diagonal elements are non-positive.*

(ii) A square matrix A of class Z is said to belong to class K if there exits a matrix $C \succeq 0$ and a number $k > \rho(C)$ such that $A = kI - C$.

(iii) A square matrix A of class Z is said to belong to class K_o if there exists a matrix $C \succeq 0$ and a number $k \geq \rho(C)$ such that $A = kI - C$.

Matrices of class K and K_o are called M-matrices (Metzler matrices). Note that equivalent definitions of matrices of class K and K_o are provided in [15]. We briefly recall that a matrix A is irreducible if and only if its associated directed graph is strongly connected. The following theorem presents a few useful properties of these matrices.

Theorem 1. *(i) If $A \in Z$ and if there exists $x \succ 0$ such that $Ax \succeq 0$, then $A \in K_o$.*

(ii) If $A \in \mathbb{R}^{n \times n}$ belongs to the class K_o, is irreducible and singular, then there exists $u \succ 0$ such that $Au = 0$. Moreover, $\text{rank}(A) = n - 1$.

(iii) If $A \in K_o$, then every eigenvalue of A has nonnegative real part.

Theorem 1 is proved in Theorem 5.11, 5.8 and 5.3 in [15], respectively.

Next we present a novel application of these concepts. For $\alpha \in]0,1]$, let $k_i \in [\alpha, 1]$ for all $i \in \{1, \dots, n\}$, and define the $n \times n$ square matrices

$$A(k_1, \dots, k_n) = \begin{bmatrix} -k_1 - k_2 & k_2 & 0 & \dots & k_1 \\ k_2 & -k_2 - k_3 & k_3 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ k_1 & 0 & \dots & k_n & -k_n - k_1 \end{bmatrix},$$

and $B(k_1, \dots, k_n) = \big(I_n + A(k_1, \dots, k_n)\big)^2 - I_n$. These matrices have the following useful properties.

Lemma 2. *The matrices $A(k_1, \dots, k_n)$ and $B(k_1, \dots, k_n)$ have rank $n-1$ and their eigenvalues have non-positive real part.*

Proof. Let $M = -A(k_1, \dots, k_n)$ so that $M \in Z$. By Theorem 1(i) with $x = \mathbf{1}$, it can be seen that $M \in K_o$. Moreover M is irreducible because represents a strongly connected graph, and singular because $M\mathbf{1} = 0$. We can then apply Theorem 1(ii) which proves that the origin is an eigenvalue of M with multiplicity 1. With Theorem 1(iii) finally it can be proved that M is positive semidefinite which implies that $A(k_1, \dots, k_n)$ is negative semidefinite. The same proof applies to $B(k_1, \dots, k_n)$. □

3 Interpolation with global information

In this first section we propose and analyze an algorithm that uniformly distributes the agents according to the pseudo-distance introduced in the previous section. We assume that the agents have complete knowledge of the contour of a body they have to approximate. The approximation set is obtained by linear interpolation of the positions of the agents, hence it will be called interpolating polygon. We prove convergence when the contour is time invariant.

3.1 Problem setup and notation

Let $\|v\|$ denote the Euclidian norm of $v \in \mathbb{R}^n$. If v is a scalar, then $|v|$ denotes its absolute value. Let $\bar{\mathbb{R}}_+$ be the set of non negative real numbers and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Let ∂Q be the boundary of a connected, and possibly non-convex set Q in $\mathbb{R}^2$. Let $\gamma\colon [0,1] \to \mathbb{R}^2$ be a parametric representation of the boundary; we assume that $\gamma'(s) \neq 0$ for all $s \in [0,1]$, that $\gamma(0) = \gamma(1)$, and that s increases as we traverse the curve in the counterclockwise direction. We define the curvature $\kappa\colon [0,1] \to \bar{\mathbb{R}}_+$ of the curve γ by

$$\kappa(s) = \frac{\|\gamma'(s) \times \gamma''(s)\|}{\|\gamma'(s)\|^3}.$$

Let $p_1, \dots, p_n \in \mathbb{R}^2$ be the locations of n robots. Let $\pi_i \in \partial Q \subset \mathbb{R}^2$ be the point on the boundary closest to the robot p_i, i.e.,

$$\pi_i = \mathrm{argmin}_{z \in \partial Q} \|p_i - z\|,$$

and let $s_i \in [0,1]$ satisfy $\gamma(s_i) = \pi_i$ (see Figure 1). The projection π_i might not being unique for some ∂Q and some positions p_i. We choose π_i randomly among all possible projections of p_i. Let $L\colon \partial Q \times \partial Q \to \bar{\mathbb{R}}_+$ be the arc length along the boundary ∂Q between two points measured in the counterclockwise direction starting from the first argument, i.e.,

$$L(\pi_1, \pi_2) = \int_{s_1}^{s_2} \|\gamma'(s)\| ds,$$

where $\pi_k = \gamma(s_k)$, $k \in \{1, 2\}$. Given $\lambda \in \mathbb{R}^+$, let $D_\lambda \colon \partial Q \times \partial Q \to \bar{\mathbb{R}}_+$, be the pseudo-distance defined by:

$$D_\lambda(\pi_i, \pi_j) = \lambda \int_{s_i}^{s_j} \kappa^{1/3}(s) \|\gamma'(s)\| ds + (1 - \lambda) L(\pi_i, \pi_j),$$

where $\pi_k = \gamma(s_k)$, $k \in \{i, j\}$. Occasionally on we will use the identifications $L(s_i, s_j) \equiv L(\pi_i, \pi_j)$, and $D_\lambda(s_i, s_j) \equiv D_\lambda(\pi_i, \pi_j)$, where $\pi_k = \gamma(s_k)$, $k \in \{i, j\}$.

Now, given n points $\{\pi_1, \ldots, \pi_n\} \in \partial Q$ we can construct the Voronoi partition of ∂Q based on the pseudo-distance D_λ. For a point π_i we define u_i the position of its immediate counterclockwise neighbor as follows: $u_i = \gamma(s_i^u) = \pi_j$, where $j = \operatorname{argmin}_{q \in \{1,\ldots,n\}} L(\pi_i, \pi_q)$. Analogously, we define ℓ_i as the position of its immediate clockwise neighbor as follows. $\ell_i = \gamma(s_i^\ell) = \pi_j$, where $j = \operatorname{argmin}_{q \in \{1,\ldots,n\}} L(\pi_q, \pi_i)$.

We associate any agent with its Voronoi cell $V_i = \{\gamma(s) \,|\, s \in [m_i^\ell, m_i^u]\}$ where m_i^ℓ and m_i^u are defined implicitly as follows: $D_\lambda(s_i^\ell, m_i^\ell) = D_\lambda(m_i^\ell, s_i)$ and $D_\lambda(s_i, m_i^u) = D_\lambda(m_i^u, s_i^u)$. Furthermore, we let $\mu_i^u = \gamma(m_i^u)$ and $\mu_i^\ell = \gamma(m_i^\ell)$. For any Voronoi cell we define its center $C_i = \gamma(s_i^*)$, where s_i^* is implicitly defined by $D_\lambda(m_i^\ell, s_i^*) = D_\lambda(s_i^*, m_i^u)$. Finally, we define the set of the neighbors of agent i according to a disk graph: $\mathcal{N}_i = \{j \in \{1, \ldots, n\} | \ \|p_j - p_i\| \leq r\}$. All the quantities just defined are illustrated in Figure 1.

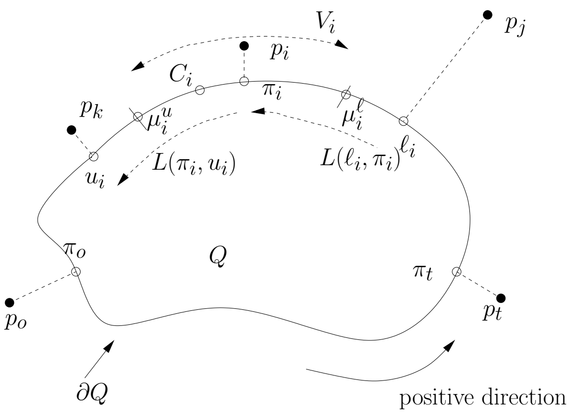


Figure 1: In this figure p_i, p_j, p_k, p_o, and p_t and their projections, π_i, ℓ_i, u_i, π_o, and π_t onto the boundary ∂Q are illustrated. There are also indicated the midpoints μ_i^ℓ and μ_i^u, according to the pseudo-distance D_λ between ℓ_i and π_i, and between π_i and u_i. V_i is the Voronoi cell associated with agent i, and the Voronoi center is C_i. $L(\ell_i, \pi_i)$ and $L(\pi_i, u_i)$ are the arc length of the portion of ∂Q between ℓ_i and π_i, and between π_i and u_i.

3.2 Curvature-weighted Deployment Algorithm

In this section we present and analyze an algorithm that will allow n robots to be equally distributed, according to the pseudo-distance D_λ, along the boundary of Q. In other words, the objective is to distribute the robots in such a way that their Voronoi cell has pseudo-length equal to

$$D_\lambda^* = \frac{1}{n} \int_0^1 \left(\lambda \kappa^{1/3}(s) + (1 - \lambda) \right) \|\gamma'(s)\| ds.$$

The algorithm is summarized in the following table.

Name: CURVATURE-WEIGHTED DEPLOYMENT ALGORITHM
Goal: Uniformly distribute the agents according to the pseudo-distance D_λ.
Data: Parametric representation of the boundary, γ.

```
Between two communication rounds, local agent $i \in \{1,\dots,n\}$ performs:
 1: Calculate its projection $\pi_i$.
 2: Acquire the projections $\pi_j$ for $j \in \mathcal{N}_i$.
 3: if $\mathcal{N}_i \neq \emptyset$, then
 4:    determine its immediate counterclockwise and clockwise neighbors, i.e., set
           $\ell_i := \pi_j, \quad \text{where } j := \operatorname{argmin}_{q\in\mathcal{N}_i} L(\pi_q, \pi_i),$
           $u_i := \pi_j, \quad \text{where } j := \operatorname{argmin}_{q\in\mathcal{N}_i} L(\pi_i, \pi_q),$
       and calculate the center of its Voronoi cell $C_i$,
 5: else
 6:    $C_i := \pi_i$.
 7: end if
 8: if $p_i \notin \partial Q$, i.e., the agent is not on the boundary, and $C_i \neq \pi_i$, then
 9:    first move towards $\pi_i$ then towards $C_i$, following the tangent to $\partial Q$ at $\pi_i$.
10: end if
11: if $p_i \notin \partial Q$, i.e., the agent is not on the boundary, and $C_i = \pi_i$, then
12:    move towards $\pi_i$.
13: end if
14: if $p_i \in \partial Q$, then
15:    move towards $C_i$ sliding along the boundary $\partial Q$.
16: end if
```

Theorem 3. *If the communication radius r is larger than the diameter of Q, then the* CURVATURE-WEIGHTED DEPLOYMENT ALGORITHM *guarantees that, for all $i \in \{1,\dots,n\}$,*

$$\lim_{t\to+\infty} D_\lambda(s_i, s_{i+1}) = \lim_{t\to+\infty} D_\lambda(s_i^*, s_{i+1}^*) = D_\lambda^*. \tag{1}$$

Proof. The algorithm assures that the agents will reach the boundary at some point, so without loss of generality we will assume that $p_i = \pi_i \in \partial Q$ for all $i \in \{1,\dots,n\}$.

The first equality in equation (1) is true because the agents always move towards their Voronoi center so we have to prove only the second equality. By hypothesis r is larger than the diameter of Q, this guarantees that $\mathcal{N}_i = \{1,\dots,n\} \setminus \{i\}$ for all $i \in \{1,\dots,n\}$. Furthermore $\{\ell_i, u_i\}$ are the correct neighbors of agent i for all $i \in \{1,\dots,n\}$ in the ring topology graph. Then, without loss of generality, we can relabel all the agents so that they are in increasing order if we move along the boundary in the counterclockwise direction. Hence, agents $i+1$ and $i-1$ are the clockwise and counterclockwise neighbors of agent i for all $i \in \{1,\dots,n\}$. This means that $\pi_{i+1} \equiv u_i$ and $\pi_{i-1} \equiv \ell_i$; we use the convention $\pi_{n+1} = \pi_1$, and $\pi_0 = \pi_n$. Let $\mu_i^\ell = \gamma(m_i^\ell)$ be the midpoint between the projections π_{i-1} and π_i according to the pseudo-distance $D_\lambda(\pi_{i-1}, \pi_i)$, then

$$D_\lambda(\pi_{i-1}, \mu_i^\ell) = D_\lambda(\mu_i^\ell, \pi_i) = \frac{1}{2} D_\lambda(\pi_{i-1}, \pi_i). \tag{2}$$

Analogously, let $\mu_i^u = \gamma(m_i^u)$ be the midpoint between the projections π_i and π_{i+1} according to the pseudo-distance $D_\lambda(\pi_i, \pi_{i+1})$, then

$$D_\lambda(\pi_i, \mu_i^u) = D_\lambda(\mu_i^u, \pi_{i+1}) = \frac{1}{2} D_\lambda(\pi_i, \pi_{i+1}). \tag{3}$$

By definition of Voronoi center C_i we have:

$$D_\lambda(\mu_i^\ell, C_i) = D_\lambda(C_i, \mu_i^u) = \frac{D_\lambda(\mu_i^\ell, \pi_i) + D_\lambda(\pi_i, \mu_i^u)}{2}. \tag{4}$$

Substituting (2) and (3) in (4):

$$D_\lambda(\mu_i^\ell, C_i) = D_\lambda(C_i, \mu_i^u) = \frac{D_\lambda(\pi_{i-1}, \pi_i) + D_\lambda(\pi_i, \pi_{i+1})}{4}. \tag{5}$$

The pseudo-distance between C_i and π_i can be now calculated as follows:

$$D_\lambda(C_i, \pi_i) = D_\lambda(\mu_i^\ell, \pi_i) - D_\lambda(\mu_i^\ell, C_i), \quad \text{if } C_i \text{ is encountered when going from } \mu_i^\ell \text{ to } \pi_i, \tag{6}$$

$$D_\lambda(\pi_i, C_i) = D_\lambda(\pi_i, \mu_i^u) - D_\lambda(C_i, \mu_i^u), \quad \text{if } C_i \text{ is encountered when going from } \pi_i \text{ to } \mu_i^u. \tag{7}$$

Equations (6) and (7) refer to the two different situations described in Figure 2.

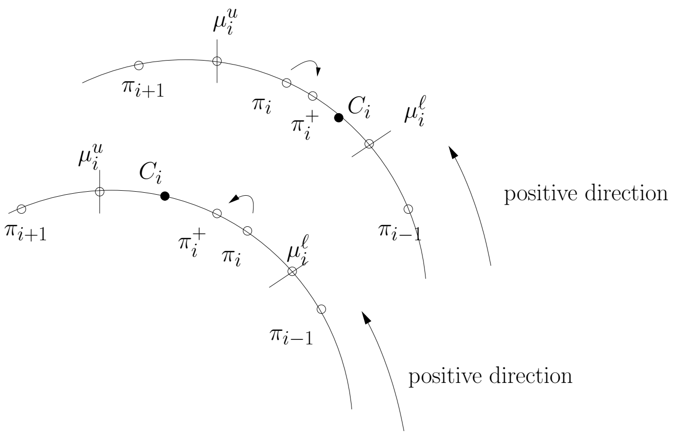


Figure 2: The center C_i of the Voronoi cell, associated with agent i, can be to the left (upper arc) or to the right (lower arc) of the projection π_i of the point p_i.

Recalling (2), (3), and (5), (6) and (7) become:

$$D_\lambda(C_i, \pi_i) = \frac{D_\lambda(\pi_{i-1}, \pi_i) - D_\lambda(\pi_i, \pi_{i+1})}{4}, \quad \text{if } C_i \text{ is encountered when going from } \mu_i^\ell \text{ to } \pi_i, \tag{8}$$

$$D_\lambda(\pi_i, C_i) = \frac{-D_\lambda(\pi_{i-1}, \pi_i) + D_\lambda(\pi_i, \pi_{i+1})}{4}, \quad \text{if } C_i \text{ is encountered when going from } \pi_i \text{ to } \mu_i^u. \tag{9}$$

According to the algorithm, agent i slides along the boundary with constant velocity towards C_i and it reaches the position π_i^+. The pseudo-distance D_λ between the position of agent i between two communication rounds can be expressed as follows:

$$D_\lambda(\pi_i^+, \pi_i) = k_i \frac{D_\lambda(\pi_{i-1}, \pi_i) - D_\lambda(\pi_i, \pi_{i+1})}{4}, \quad \text{if } C_i \text{ is encountered when going from } \mu_i^\ell \text{ to } \pi_i, \tag{10}$$

$$D_\lambda(\pi_i, \pi_i^+) = k_i \frac{-D_\lambda(\pi_{i-1}, \pi_i) + D_\lambda(\pi_i, \pi_{i+1})}{4}, \quad \text{if } C_i \text{ is encountered when going from } \pi_i \text{ to } \mu_i^u, \tag{11}$$

where $\alpha \le k_i \le 1$, with α being the uniform lowerbound on k_is only after the agents reach the boundary.

In the same way we can calculate $D_\lambda(\pi_{i-1}^+, \pi_{i-1})$ or $D_\lambda(\pi_{i-1}, \pi_{i-1}^+)$ according to the relative position between π_{i-1}^+ and π_{i-1}:

$$D_\lambda(\pi_{i-1}^+, \pi_{i-1}) = k_{i-1} \frac{D_\lambda(\pi_{i-2}, \pi_{i-1}) - D_\lambda(\pi_{i-1}, \pi_i)}{4}, \tag{12}$$

$$D_\lambda(\pi_{i-1}, \pi_{i-1}^+) = k_{i-1} \frac{-D_\lambda(\pi_{i-2}, \pi_{i-1}) + D_\lambda(\pi_{i-1}, \pi_i)}{4}. \tag{13}$$

We can now express the new pseudo-distance between π_{i-1}^+ and π_i^+ as a function of $D_\lambda(\pi_{i-1}, \pi_i)$. As shown in Figure 3, there four possible cases to take into account. For each of them $D_\lambda(\pi_{i-1}^+, \pi_i^+)$ is expressed as follows:

$$D_\lambda(\pi_{i-1}^+, \pi_i^+) = \begin{cases} D_\lambda(\pi_{i-1}, \pi_i) + D_\lambda(\pi_i, \pi_i^+) - D_\lambda(\pi_{i-1}, \pi_{i-1}^+), & \text{case I ,} \\ D_\lambda(\pi_{i-1}, \pi_i) - D_\lambda(\pi_i^+, \pi_i) + D_\lambda(\pi_{i-1}^+, \pi_{i-1}), & \text{case II ,} \\ D_\lambda(\pi_{i-1}, \pi_i) + D_\lambda(\pi_i, \pi_i^+) + D_\lambda(\pi_{i-1}^+, \pi_{i-1}), & \text{case III ,} \\ D_\lambda(\pi_{i-1}, \pi_i) - D_\lambda(\pi_i^+, \pi_i) - D_\lambda(\pi_{i-1}, \pi_{i-1}^+), & \text{case IV .} \end{cases} \tag{14}$$

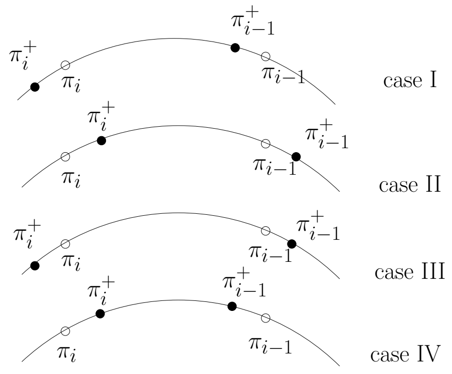


Figure 3: In this figure the four possible relative positions between π^+, π_i, π_{i-1}^+, and π_{i-1} are shown.

Substituting (10), (11), (12), and (13) in (14):

$$D_\lambda(\pi_{i-1}^+, \pi_i^+) = \frac{k_{i-1}}{4} D_\lambda(\pi_{i-2}, \pi_{i-1}) + (1 - \frac{k_{i-1}}{4} - \frac{k_i}{4}) D_\lambda(\pi_{i-1}, \pi_i) + \frac{k_i}{4} D_\lambda(\pi_i, \pi_{i+1}), \tag{15}$$

for all the four cases. If we now call $\mathbf{D}$ the column vector obtained by stacking up the pseudo-distances $D_\lambda(\pi_{i-1}, \pi_i)$ for all $i \in \{1, \dots, n\}$, and $\mathbf{D}^+$ its update, then:

$$\mathbf{D}^+ = (I_n + A(k_1/4, \dots, k_n/4))\mathbf{D}, \tag{16}$$

where $A(k_1/4, \dots, k_n/4)$ was defined in Section 2.

We need to be able to express α, lower bound of k_is, in order to apply the results of [11]. Since the vectors $\gamma'(s)$, and $\kappa(s)$ are defined on a compact set, their modulus has a maximum and a minimum value. Let $\|\gamma'_{\max}\|$, $\|\gamma'_{\min}\|$, $\|\kappa_{\max}\|$, and $\|\kappa_{\min}\|$ be the maximum and minimum values of the norm of $\gamma'(s)$ and $\kappa(s)$. Let us suppose now that agent i has to slide along the boundary ∂Q from the position π_i to position π_i^+ at speed v in the sampling time $\Delta t > 0$, then:

$$v = \frac{L(\pi_i, \pi_i^+)}{\Delta t} = \frac{\int_{s_i}^{s_i^+} \|\gamma'\| ds}{\Delta t} \implies v \leq \frac{\|\gamma'_{\max}\|(s_i^+ - s_i)}{\Delta t} \implies (s_i^+ - s_i) \geq \frac{v \Delta t}{\|\gamma'_{\max}\|}. \tag{17}$$

We can lower bound and upper bound the quantity $D_\lambda(\pi_i, \pi_i^+)$ recalling (11):

$$\frac{k_i}{4} D_\lambda(\partial Q) \geq D_\lambda(\pi_i, \pi_i^+) = \int_{s_i}^{s_i^+} \lambda \kappa^{\frac{1}{3}} \|\gamma'\| + (1-\lambda)\|\gamma'\| ds \geq \left[\lambda \kappa_{\min}^{\frac{1}{3}} \|\gamma'_{\min}\| + (1-\lambda)\|\gamma'_{\min}\|\right](s_i^+ - s_i).$$

We can write:

$$\frac{k_i}{4} D_\lambda(\partial Q) \geq \left[\lambda \|\kappa_{\min}\|^{\frac{1}{3}} + (1-\lambda)\right] \|\gamma'_{\min}\| (s_i^+ - s_i) \geq \left[\lambda \|\kappa_{\min}\|^{\frac{1}{3}} + (1-\lambda)\right] \|\gamma'_{\min}\| \left(\frac{v\Delta t}{\|\gamma'_{\max}\|} \right).$$

The lowerbound on k_i can be evaluated as follows:

$$k_i \geq 4 \frac{\lambda \|\kappa_{\min}\|^{\frac{1}{3}} + (1-\lambda)}{D_\lambda(\partial Q)} \frac{\|\gamma'_{\min}\|}{\|\gamma'_{\max}\|} v\Delta t = \alpha.$$

Given the boundary ∂Q, the speed, and the sampling time, α can be calculated. If the speed v is very large then α could be larger than 1. This means that any agent can move in a sampling interval from a point on the boundary to any other point, then the lowerbound on k_i is, as the upperbound, equal to 1.

We can now use Theorem 3 of [11] and the fact that the dynamic matrix $A(k_1, \ldots, k_n)$ in equation (16) is doubly stochastic to complete the proof. □

Remark 4. *Instead of relying on the results in [11], we sketch here an alternative proof given by a direct Lyapunov argument. If $V(\mathbf{D}) = (\mathbf{D} - D^*_\lambda \mathbf{1})^T(\mathbf{D} - D^*_\lambda \mathbf{1})$, then*

$$V(\mathbf{D}^+) - V(\mathbf{D}) = (\mathbf{D}^+)^T(\mathbf{D}^+) - \mathbf{D}^T\mathbf{D} = \mathbf{D}^T B(k_1/4, \ldots, k_n/4)\mathbf{D} \leq 0,$$

where $B(k_1/4, \ldots, k_n/4)$ is defined in Section 2. Recall that $\mathbf{1}^T\mathbf{D} = \mathbf{1}\mathbf{D}^+ = D_\lambda(\partial Q)$ and that the dynamics matrices are symmetric. It can be proved (see Lemma 2) that all the eigenvalues of $B(k_1/4, \ldots, k_n/4)$ have strictly negative real part, except for one placed at the origin. The eigenvector associated with the eigenvalue 0 is the vector $\mathbf{1}$. Using the LaSalle Invariance Principle we conclude that consensus is reached.

Conjecture 1. *We conjecture that* Curvature-weighted Deployment Algorithm *achieves consensus even if r is smaller than the diameter of Q but larger than r^*, where r^* is defined by:*

$$r^* = \max_{\substack{e,f \in [0,1] \\ D(e,f) = D^*}} \int_e^f \|\gamma'(s)\| ds.$$

The lowerbound r^* guarantees that, if consensus is reached, each agent can communicate with its immediate clockwise and counterclockwise neighbors (according to the ring graph, i.e., the neighbors along the boundary). However, before consensus is reached, a possible situation, illustrated in Figure 4, is one in which some agents cannot communicate with their immediate neighbors and some agents are isolated. Each non isolated agent will move in such a way that the pseudo-distance between itself and its clockwise neighbor, and between itself and its counterclockwise neighbor have the same value. As a consequence, the pseudo-distance between an agents and those that it recognizes as its neighbors will grow and potentially will be larger than D^*_λ. In doing so it might get to see its correct neighbors or it might get isolated because, at a certain point, the euclidian distance between itself and its last neighbors is larger than r^*. Since the boundary ∂Q is a closed set, not every agent can be isolated and there will always be some agent that can detect other agents and move accordingly. It is our conjecture that, eventually, every agent will determine its correct immediate neighbors according to the ring topology graph, and hence consensus will be reached.

3.3 Simulations

In this section we present the implementation results of the Curvature-weighted Deployment Algorithm when ∂Q is time-invariant and described by, for $\theta \in [0, 1)$,

$$\gamma(\theta) = \Big(2 + \cos(10\pi\theta) + 0.5\sin(4\pi\theta)\Big) \begin{bmatrix} \cos(2\pi\theta) \\ \sin(2\pi\theta) \end{bmatrix}. \tag{18}$$

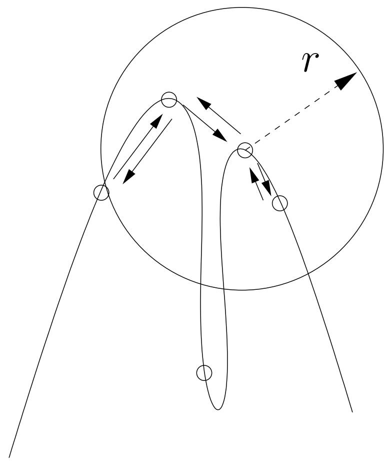


Figure 4: In this figure four agents are on a portion of the boundary ∂Q. The radius of communication is r. One agent is isolated, while the other three have at least a neighbor according to the disk graph. With knowledge about their neighbors in the disk graph, the agents are not able to correctly identify their neighbors along the boundary.

In Figure 5 five different frames of the simulation are collected. The frames show for $t = 0$, $t = 0.25$, $t = 1$, $t = 5$, and $t = 25$ the positions of agents and the boundary ∂Q. The value of λ is $\frac{10}{11}$. The sampling time is $t = 0.05$ seconds while the radius of the communication disk is $r = 3$.

In Figure 6 it is shown the error $\max_{i\in\{1,\dots,n\}} D_\lambda(p_i, p_{i+1}) - \min_{i\in\{1,\dots,n\}} D_\lambda(p_i, p_{i+1})$ in the consensus on D_λ reached by the agents. The difference should vanish but it does not because of numerical errors. For example, because the Voronoi center is defined implicitly, the agents need to estimate it using the bisection method and this, of course, introduces addtional error in the evaluation of Voronoi cell and Voronoi center.

4 Interpolation with local information

The Curvature-weighted Deployment Algorithm was developed under the assumption that all the agents have complete knowledge of the boundary ∂Q. Here we consider a more generic and realistic scenario in which every sensing agent has only local knowledge of the boundary to be reconstructed. We suppose that the sensing agents can locally estimate the tangent and the curvature of ∂Q. For the case of UAVs surveilling a visible boundary, this information could be provided by a camera and an edge detection algorithm. Another possibility is to substitute every agent with a formation of chemical sensors. In the recent work [6] the authors propose an optimal formation of four agents to estimate the gradient and the curvature of a given level set in a field. We assume that an initial estimate of the boundary is available so that the interpolation points can be distributed (possibly non uniformly) on the boundary and the pseudo-distance $\widehat{D}_\lambda$ between any two neighbors is known. We assume also that every agent is equipped with wireless communication devices to communicate with a data fusion center.

To interpolate the unknown slowly time-varying boundary ∂Q, we introduce a counterclockwise ordered set of interpolation points $\{p_1, \dots, p_{n_{\text{ip}}}\}$. These are virtual positions stored in a data fusion center together with the tangent of ∂Q at all interpolation points, and the pseudo-distance between any interpolation point and its counterclockwise neighbor. The agents have two objectives: (i) update the interpolation points such that they are uniformly distributed along ∂Q according to the estimated pseudo-distance $\widehat{D}_\lambda$, (ii) be equally distributed along the boundary according to arc length distance.

To achieve these objectives we propose a novel Estimate Update and Pursuit Algorithm

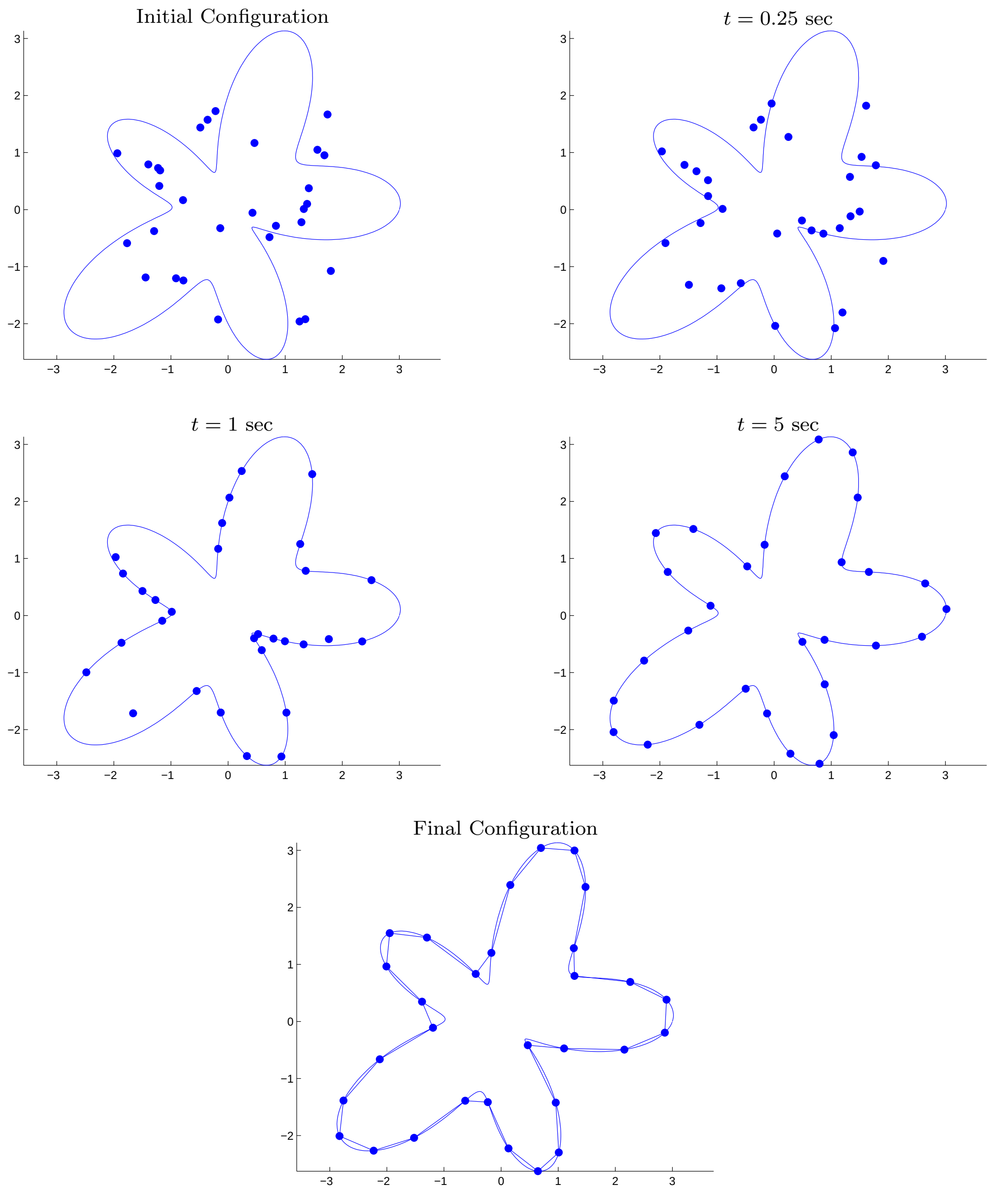


Figure 5: This figure shows five different instants of 25 seconds simulation obtained by implementing the Curvature-weighted Deployment Algorithm with $n = 30$, $\lambda = \frac{10}{11}$, $r = 3$.

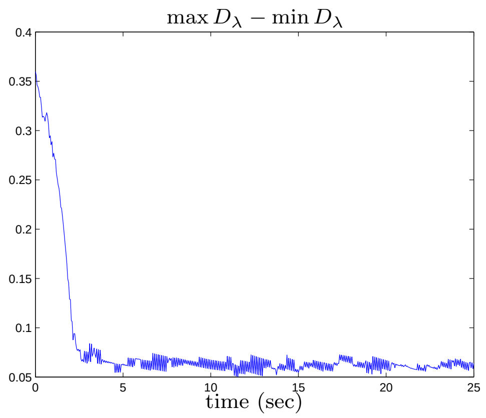


Figure 6: CURVATURE-WEIGHTED DEPLOYMENT ALGORITHM. The plot shows the error $\max_{i\in\{1,\dots,n_{\mathrm{ip}}\}} D_\lambda(p_i,p_{i+1}) - \min_{i\in\{1,\dots,n_{\mathrm{ip}}\}} D_\lambda(p_i,p_{i+1})$ vs time.

that can be summarized as follows. After reaching a point on ∂Q, the sensing agents move along ∂Q to collect the following data: (i) points belonging to ∂Q, and (ii) tangent and curvature of ∂Q at those points. Using these measurements and communicating with the data center, they complete the following three steps. In the first step, they determine which interpolation point p_i they are closer to and then project it onto the measured boundary. In the second step, they adjust p_{i-1} so that it is at the center of its Voronoi cell along ∂Q. In the third step, they estimate the arc length of distance between them and their immediate clockwise and counterclockwise neighbors and use this information to speed up or slow down. The first two steps have the combined effect of updating the local estimates of the boundary. Thanks to the third step, the agents distribute themselves uniformly along the boundary.

In what follows we present the ESTIMATE UPDATE AND PURSUIT ALGORITHM in some detail and we analyze its stability.

4.1 Problem setup and notation

Let $\gamma\colon \overline{\mathbb{R}}_+ \times [0,1] \to \mathbb{R}^2$ be a parametric representation of the time-varying boundary so that, at fixed $t \in \overline{\mathbb{R}}_+$ and for all $s \in [0,1]$, $\gamma(t,s)$ describes the boundary at time t. We assume that $\frac{\partial\gamma(t,s)}{\partial s} = \gamma'(t,s) \neq 0$ for all $s \in [0,1]$ and for all t, that $\gamma(t,0) = \gamma(t,1)$, and that s increases as we traverse the curve in the counterclockwise direction. We also assume that $\gamma(t,s)$ is smooth with respect to s and t and that the length of the boundary ∂Q is upperbounded and lowerbounded uniformly in t.

Let $p_1,\dots,p_{n_{\mathrm{ip}}} \in \mathbb{R}^2$ be the locations of n_{ip} ordered interpolation points. Let $P_i(t)$, with $i \in \{1,\dots,n_{\mathrm{a}}\}$ and $n_{\mathrm{ip}} \gg 3n_{\mathrm{a}}$, be the positions of the sensing agents at a given time. Both the interpolation points and the sensing agents are ordered counterclockwise. We assume that the sensing agents move counterclockwise along the boundary, with speed v_i.

We assume that each agent maintains some variables in its memory that are described as follows. The state of the sensing agent i is:

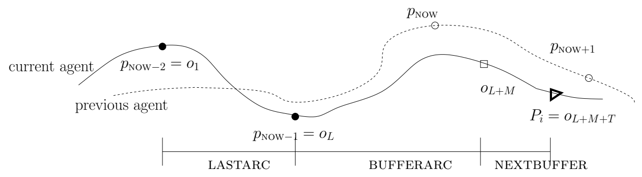
$$\{\mathrm{NOW}^i, \mathrm{LASTARC}^i, \mathrm{BUFFERARC}^i, \mathrm{NEXTBUFFER}^i\},$$

where the first variable is a counter, and the other three are a discrete representation of the subset of ∂Q the sensing agent is flying over. For simplicity we will omit the upperscript and lowerscript i and we will introduce them when necessary.

Let $\textsc{now} \in \{1, \dots, n_{\mathrm{ip}}\}$ be the next point to be projected onto ∂Q. Let $\mathcal{O} = \textsc{lastarc} \cup \textsc{bufferarc} \cup \textsc{nextbuffer} \subset \partial Q$ be the set of observations collected by the sensing agent up to time t while going from $p_{\textsc{now}-2}$ towards $p_{\textsc{now}}$ along ∂Q defined as follows:

$$\begin{aligned} \textsc{lastarc} &= \{o_1, \dots, o_L\}, \\ \textsc{bufferarc} &= \{o_{L+1}, \dots, o_{L+M}\}, \\ \textsc{nextbuffer} &= \{o_{L+M+1}, \dots, o_{L+M+T}\}, \end{aligned} \tag{19}$$

where $L, M \in \mathbb{N}$, $T \in \mathbb{N}_0$, $o_1 = p_{\textsc{now}-2}$, $o_L = p_{\textsc{now}-1}$, and $o_{L+M+T} = P(t)$. The following figure illustrates these notation and quantities. The solid line represents ∂Q as seen by the agent i, while the dashed line represents ∂Q as seen by the agent $i-1$. The sensing agent is represented by a triangle. The white circles represent the interpolation points before the agent updates them, whereas the black circles represent the interpolation points after the agent has updated them. The square represents the last point belonging to $\textsc{bufferarc}$. The first and the last point of the three data structures $\textsc{lastarc}$, $\textsc{bufferarc}$, and $\textsc{nextbuffer}$ are shown, the others are omitted for clarity.



Before defining the point o_{L+M} and the index M we introduce the set of estimated tangent vectors to ∂Q, $\widehat{\gamma'} \colon \mathcal{O} \to \mathbb{R}^2$, and the set of estimated curvature of ∂Q, $\widehat{\kappa} \colon \mathcal{O} \to \bar{\mathbb{R}}_+$. In other words, $\widehat{\gamma'}(o_j)$ and $\widehat{\kappa}(o_j)$ are estimated tangent vector and curvature at the point o_j, for $j \in \{1, \dots, L+M+T\}$. We can now define $\widehat{D}_\lambda \colon \mathcal{O} \times \mathcal{O} \to \bar{\mathbb{R}}_+$ as the discretized pseudo-distance between two observations o_j and o_h, with $h, j \in \{1, \dots, L+M+T\}$, and $h > j$. We shall characterize implicitly the observation o_{L+M} as follows

$$\widehat{D}_\lambda(o_{L+1}, o_{L+M}) = 2\widehat{D}_\lambda^-(p_{\textsc{now}-1}, p_{\textsc{now}}),$$

where $\widehat{D}_\lambda^-(p_{\textsc{now}-1}, p_{\textsc{now}})$ is the estimated pseudo-distance between $p_{\textsc{now}-1}$ and $p_{\textsc{now}}$ when an agent updated for the last time $p_{\textsc{now}}$. This information is assumed to be stored in the data center.

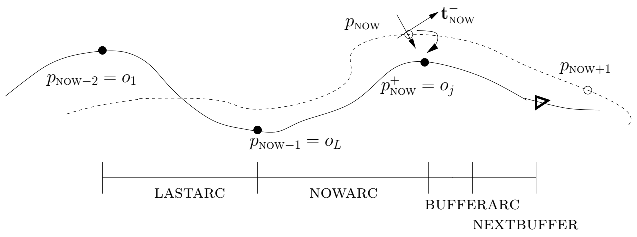
The sets $\textsc{lastarc}$, and $\textsc{bufferarc}$ will be used by the sensing agents to define the projection onto ∂Q of $p_{\textsc{now}}$ and the Voronoi center $\widehat{C}_{\textsc{now}-1}$ of the interpolation point $p_{\textsc{now}-1}$. We recall that the positions $o_1, \dots, o_{L+M+T}$ are points on the plane that the sensing agent has visited in previous instants while moving along the boundary ∂Q, i.e., $o_j = P(\tau)$ for some $\tau < t$ and for all $j \in \{1, \dots, L+M+T\}$. We can say that the points $o_1, \dots, o_{L+M+T}$ are a fine discretization of the portion of ∂Q from $p_{\textsc{now}-2}$ to the current position of the sensing agent P, while the indices $p_{\textsc{now}-2}$, $p_{\textsc{now}-1}$, and $p_{\textsc{now}}$ are a coarser discretization of the same arc.

To calculate the Voronoi cell $\widehat{V}_{\textsc{now}-1}$ along ∂Q of the interpolation point $p_{\textsc{now}-1}$, we first need to project $p_{\textsc{now}}$ on ∂Q. Let $o_{\bar{j}}$ be the projection of $p_{\textsc{now}}$, defined by:

$$p_{\textsc{now}}^+ := o_{\bar{j}} = \operatorname{argmin}_{o_j \in \textsc{bufferarc}} \|(o_j - p_{\textsc{now}}) \cdot \mathbf{t}_{\textsc{now}}^-\|,$$

where $\mathbf{t}_{\textsc{now}}^- = \frac{\widehat{\gamma'}(p_{\textsc{now}})}{\|\widehat{\gamma'}(p_{\textsc{now}})\|}$ is the unit-length tangent vector at $\partial Q(t^-)$ at the interpolation point $p_{\textsc{now}}$ last time the interpolation point was updated. In other words the projection of $p_{\textsc{now}}$ at time t^+ on $\partial Q(t^+)$ is the intersection of $\partial Q(t^+)$ with the normal vector to $\partial Q(t^-)$ at $p_{\textsc{now}}$ at time t^-. This projection is univocally defined and has the following properties. If ∂Q is time-invariant then $p_{\textsc{now}}^- = p_{\textsc{now}}^+$, if ∂Q is slowly time-varying then $p_{\textsc{now}}^+$ is close to the orthogonal projection of $p_{\textsc{now}}^-$ onto $\partial Q(t^+)$. We can now define the set $\textsc{nowarc}$

and update BUFFERARC as follows: NOWARC $= \{o_{L+1}, \ldots, o_{\bar{j}}\}$, BUFFERARC = BUFFERARC $\setminus \{o_{L+1}, \ldots, o_{\bar{j}}\}$. In the following figure, the agent (i) projects the interpolation point p_{NOW} onto $\partial Q(t)$, (ii) update the state variable BUFFERARC and generate the variable NOWARC.



Using the collected data, the sensing agents can numerically evaluate the pseudo-distances $\widehat{D}_\lambda(p_{\text{NOW}-2}, p_{\text{NOW}-1})$ and $\widehat{D}_\lambda(p_{\text{NOW}-1}, p_{\text{NOW}})$ between $p_{\text{NOW}-2}$ and $p_{\text{NOW}-1}$, and between $p_{\text{NOW}-1}$ and p_{NOW}. Recall that $p_{\text{NOW}-2} = o_1$, $p_{\text{NOW}-1} = o_L$. Let then $\widehat{V}_{\text{NOW}-1} = \{o_\ell, \ldots, o_u\}$, where $o_\ell \in$ LASTARC and $o_u \in$ NOWARC are implicitly defined by:

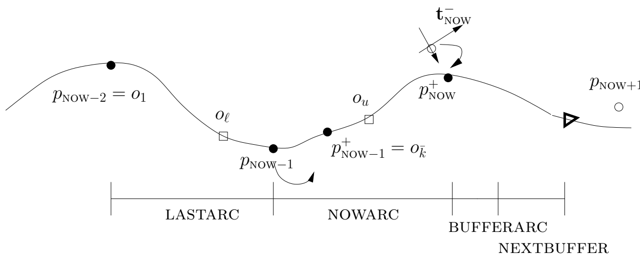
$$\widehat{D}_\lambda(o_1, o_\ell) = \widehat{D}_\lambda(o_\ell, o_L) = \frac{\widehat{D}_\lambda(o_1, o_L)}{2},$$

$$\widehat{D}_\lambda(o_L, o_u) = \widehat{D}_\lambda(o_u, o_{\bar{j}}) = \frac{\widehat{D}_\lambda(o_L, o_{\bar{j}})}{2}.$$

The point o_ℓ is the midpoint between $p_{\text{NOW}-2}$ and $p_{\text{NOW}-1}$, while o_u is the midpoint between $p_{\text{NOW}-1}$ and p_{NOW} after the latter was projected on ∂Q. We can now implicitly define the Voronoi center $\widehat{C}_{\text{NOW}-1} := o_{\bar{k}} \in$ LASTARC $\cup$ NOWARC by

$$\widehat{D}_\lambda(o_\ell, o_{\bar{k}}) = \widehat{D}_\lambda(o_{\bar{k}}, o_u) = \frac{\widehat{D}_\lambda(o_1, o_L) + \widehat{D}_\lambda(o_L, o_{\bar{j}})}{4}.$$

In the following figure, the agent (i) calculates the Voronoi cell $\widehat{V}_{\text{NOW}-1}$, and (ii) updates the interpolation point $p_{\text{NOW}-1}$ to lie optimally between $p_{\text{NOW}-2}$ and p_{NOW}.



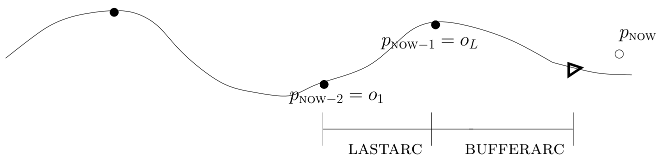
Remark 5. *In practical settings, the sensing agent might not be able to find o_ℓ, o_u, and $\widehat{C}_{\text{NOW}-1}$ that exactly satisfy the equalities above, due to the fact that the sensing agent knows only a discretization of ∂Q through the observations* LASTARC, NOWARC, *and* BUFFERARC. *To handle the discretization, one can redefine o_ℓ, o_u, and $\widehat{C}_{\text{NOW}-1}$ by:*

$$o_\ell = \text{argmin}_{o_i \in \text{LASTARC}} \left| \widehat{D}_\lambda(o_1, o_i) - \frac{\widehat{D}_\lambda(o_1, o_L)}{2} \right|,$$

$$o_u = \text{argmin}_{o_i \in \text{NOWARC}} \left| \widehat{D}_\lambda(o_i, o_L) - \frac{\widehat{D}_\lambda(o_L, o_{\bar{j}})}{2} \right|,$$

$$\widehat{C}_{\text{NOW}-1} = o_{\bar{k}} = \text{argmin}_{o_i \in \{o_\ell, \ldots, o_u\}} \left| \widehat{D}_\lambda(o_\ell, o_i) - \left(\frac{\widehat{D}_\lambda(o_L, o_{\bar{j}}) + \widehat{D}_\lambda(o_1, o_L)}{4} \right) \right|.$$

The sensing agent can now update once more the state variables as follows: $\textsc{now}^+ = \textsc{now}+1$, $\textsc{lastarc}^+ = \{o_{\bar{k}}, \dots, o_{\bar{j}}\}$, $\textsc{bufferarc}^+ = \{o_{\bar{j}+1}, \dots, o_q\}$, and $\textsc{nextbuffer}^+ = \{o_{q+1}, \dots, o_{L+M+T}\}$, where $o_q \in \textsc{bufferarc} \cup \textsc{nextbuffer}$ is implicitly defined by: $\widehat{D}_\lambda(o_{\bar{j}+1}, o_q) = 2\widehat{D}_\lambda^-(p_{\textsc{now}^+-1}, p_{\textsc{now}^+})$, if $\widehat{D}_\lambda(o_{\bar{j}+1}, o_{L+M+T}) < 2\widehat{D}_\lambda^-(p_{\textsc{now}^+-1}, p_{\textsc{now}^+})$ then $\textsc{bufferarc}^+ = \textsc{bufferarc} \cup \textsc{nextbuffer}$ and $\textsc{nextbuffer}^+ = \emptyset$. The following figure shows the state variables update as just described, in the case that $\textsc{nextbuffer} = \emptyset$.



This completes our description of the estimate update algorithm and we now focus on the pursuit objective. To uniformly distribute the sensing agents along the boundary ∂Q according to arc length, we will use the following update law for their velocities:

$$v_i(t) = v_0 + k(\widehat{L}(P_i, P_{i+1}) - \widehat{L}(P_{i-1}, P_i)),$$

with k, $v_0 > 0$ and $\widehat{L}(P_n, P_m) = \sum_{j=\textsc{now}^n+1}^{\textsc{now}^m}(\|p_{j-1} - p_j\|)$, for all $n, m \in \{1, \dots, n_\mathrm{a}\}$. Here, recall that $p_{\textsc{now}^n}, p_{\textsc{now}^{n+1}}, \dots, p_{\textsc{now}^m}$ are the interpolation points separating agent n and agent m, with $n < m$. $\widehat{L}$ is the estimated arc length of the portion of ∂Q that has to be traversed to go from the sensing agent n to the sensing agent m. The sensing agents have only local information of ∂Q but still they have to estimate the distance, along ∂Q, from their clockwise and counterclockwise neighbors in order to calculate their speed. The estimate $\widehat{L}(P_n, P_m)$ is obtained by the approximating polygon formed by the interpolation points. In practice any agent will speed up if it is closer to the agent behind it, and slow down if closer to the agent in front of it. With a saturation-like function: $\mathrm{sat}(v_i(t)) = \max\{v_{\min}, \min\{v_i(t), v_{\max}\}\}$, we will impose though that $0 < v_{\min} \le v_i(t) \le v_{\max}$ for all t.

4.2 Estimate Update and Pursuit Algorithm

In this section we present an algorithm that allows n_a sensing agents to equally distribute the n_ip interpolation points along ∂Q, according to the pseudo-distance $\widehat{D}_\lambda$. Also the algorithm uniformly distributes the n_a sensing agents along ∂Q, according to the arc length. The algorithm is summarized in the following table.

Some steps of the algorithm are affected by noise and error: i) $\widehat{\gamma'}$ and $\widehat{\kappa}$ are only estimate of the true values, ii) $\widehat{L}$ is an approximation of L, iii) the sets $\textsc{lastarc}$, $\textsc{bufferarc}$, and $\textsc{nextbuffer}$ are discretization of the subset of ∂Q that agent i is visiting, therefor, the center of the Voronoi cell of the interpolation point $p_{\textsc{now}^i-1}$ might not be calculated exactly. Let $\widehat{\mathbf{D}}(t)$ and $\mathbf{L}(t)$ be the column vectors:

$$\widehat{\mathbf{D}}(t) = \left[\widehat{D}_\lambda(p_1(t), p_2(t)), \dots, \widehat{D}_\lambda(p_{n_\mathrm{ip}-1}(t), p_{n_\mathrm{ip}}(t)), \widehat{D}_\lambda(p_{n_\mathrm{ip}}(t), p_1(t))\right]^T,$$

$$\mathbf{L}(t) = \left[L(P_1(t), P_2(t)), \dots, L(P_{n_a-1}(t), P_{n_a}(t)), L(P_{n_a}(t), P_1(t))\right]^T.$$

Consider now the disagreement vectors $\mathbf{d}(k)$ and $\delta\mathbf{L}(t)$ defined as follows:

$$\mathbf{d}(k) = \widehat{\mathbf{D}}_\lambda(k) - \frac{\mathbf{1}^T\widehat{\mathbf{D}}_\lambda(k)}{n_\mathrm{ip}}\mathbf{1}, \tag{20}$$

$$\delta\mathbf{L}(t) = \mathbf{L}(t) - \frac{\mathbf{1}^T\mathbf{L}(t)}{n_\mathrm{a}}\mathbf{1}, \tag{21}$$

note that they are orthogonal to the vector $\mathbf{1}$.

It will be proved that the dynamics of $\mathbf{d}$ and $\delta\mathbf{L}$ is input-to-state stable (ISS) where the inputs are the errors and noises above discussed. Because of the ISS property we can conclude that as long as the errors are small, the states $D_\lambda(p_i, p_{i+1})$ and $L(P_i, P_{i+1})$ will be close to the equilibrium of the unperturbed system, i.e., $D_\lambda(p_i, p_{i+1}) = D_\lambda(p_{i+1}, p_{i+2})$ for all $i \in \{1, \dots, n_{\mathrm{ip}}\}$ and $L(P_i, P_{i+1}) = L(P_{i+1}, P_{i+2})$ for all $i \in \{1, \dots, n_{\mathrm{a}}\}$.

Lemma 6. *Let $\gamma\colon [t_0, +\infty) \times [0, 1] \to \mathbb{R}^2$ describe the boundary ∂Q along time, and let $\partial\mathcal{B}(p, \epsilon)$ be the boundary of the ball centered in p and with radius ϵ. If $\gamma(t, s)$ is a smooth function of both its arguments, then $\forall\epsilon > 0\ \exists T > t_0$ such that $\forall p \in \partial Q(t_0)$ the set defined by $\partial\mathcal{B}(p, \epsilon) \cap \partial Q(t)$, $t_0 < t < T$, has only two elements.*

Lemma 6 implies that if we allow small enough changes in ∂Q, then the projection of any interpolation point p_i onto $\partial Q(t)$, as defined in the previous section, is unique.

To avoid abuse of notation, let $\overline{D}_\lambda(p_i, p_{i+1}, t)$ be the pseudo-distance along $\partial Q(t)$ between the instantaneous projection of two interpolation points onto $\partial Q(t)$. Let t_0 be the last instant of time in which the sensing agent measured $\widehat{D}_\lambda(p_i, p_{i+1})$. Then the error $|\overline{D}_\lambda(p_i, p_{i+1}, t) - \widehat{D}_\lambda(p_i, p_{i+1}, t_0)|$ is bounded and the result is stated as follows.

Lemma 7. *Let $\gamma\colon [t_0, T] \times [0, 1] \to \mathbb{R}^2$ describe the boundary ∂Q along time. If $\gamma(t, s)$ is a smooth function of both its arguments, then for $t > t_0$, $\overline{D}_\lambda(p_i, p_{i+1}, t) = \widehat{D}_\lambda(p_i, p_{i+1}, t_0) + g(t - t_0)$ for all $i \in \{1, \dots, n_{ip}\}$, where $g(t)$ is continuous and $g(0) = 0$. The quantity $\overline{D}_\lambda(p_i, p_{i+1}, t)$ denotes the pseudo-distance between two consecutive interpolation points at time t given that they were projected onto $\partial Q(t)$ from their positions at time t_0. The number $t - t_0$ is bounded from above by $\Delta t = \max_{t\in\overline{\mathbb{R}}_+} \frac{L(\partial Q(t))}{v}$, where v is the speed of the agent.*

Lemma 7 is due to the fact the the pseudo-distance $\widehat{D}_\lambda$ is the composition of smooth functions and only continuous functions in t, so it is continuous in t. Therefor, if the boundary changes slowly, then the corrective term $g(t - t_0)$ is just like a noise. If we can prove that $\widehat{D}_\lambda(p_i, p_{i+1}, t)$ gets close to the target value, we will prove that also $\overline{D}_\lambda(p_i, p_{i+1}, t)$ will get close to the target vector due to Lemma 7.

Theorem 8. *The evolution of the disagreement vectors defined by* (20) *and by* (21) *under the* Estimate Update and Pursuit Algorithm *is input-to-state stable with respect to estimation noise and deformation of the boundary $\partial Q(t)$.*

Proof. We first prove the ISS property for the dynamics of $\mathbf{d}(k)$.

Let us suppose that $\partial Q(t)$ is time-invariant, that no error affects the calculation of $\widehat{C}_{\mathrm{NOW}-1}$, the center of the Voronoi cell of the interpolation point $p_{\mathrm{NOW}-1}$ (i.e., the buffers used by the agents are continuous and not discrete). Suppose that a sensing agent has passed by the point p_{NOW}, and then it can optimally place $p_{\mathrm{NOW}-1}$. As a consequence, the pseudo-distances $\widehat{D}_\lambda(p_{\mathrm{NOW}-2}, p_{\mathrm{NOW}-1})$ and $\widehat{D}_\lambda(p_{\mathrm{NOW}-1}, p_{\mathrm{NOW}})$ will take new values that can be expressed as follows, (see Figure 7):

$$\begin{aligned}
\widehat{D}_\lambda(p_{\mathrm{NOW}-2}, p_{\mathrm{NOW}-1})^+ &= \frac{3}{4}\widehat{D}_\lambda(p_{\mathrm{NOW}-2}, p_{\mathrm{NOW}-1}) + \frac{1}{4}\widehat{D}_\lambda(p_{\mathrm{NOW}-1}, p_{\mathrm{NOW}}),\\
\widehat{D}_\lambda(p_{\mathrm{NOW}-1}, p_{\mathrm{NOW}})^+ &= \frac{1}{4}\widehat{D}_\lambda(p_{\mathrm{NOW}-2}, p_{\mathrm{NOW}-1}) + \frac{3}{4}\widehat{D}_\lambda(p_{\mathrm{NOW}-1}, p_{\mathrm{NOW}}).
\end{aligned}$$

For $i \in \{1, \dots, n_{\mathrm{ip}}\}$, define $A_i \in \mathbb{R}^{n_{\mathrm{ip}}\times n_{\mathrm{ip}}}$ by

$$(A_i)_{jk} = \begin{cases} 3/4, & \text{if } j = k = i, \text{ or } j = k = i - 1,\\ 1/4, & \text{if } j = i - 1 \text{ and } k = i, \text{ or if } j = i, \text{ and } k = i - 1,\\ \delta_{jk}, & \text{otherwise.} \end{cases}$$

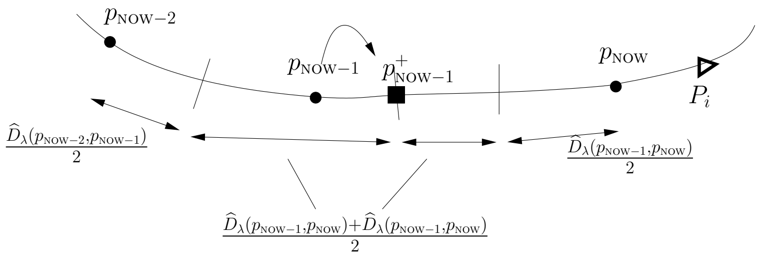

$p_{\text{NOW}-2}$
$p_{\text{NOW}-1}$
$p^+_{\text{NOW}-1}$
p_{NOW}
P_i
$\frac{\widehat{D}_\lambda(p_{\text{NOW}-2},p_{\text{NOW}-1})}{2}$
$\frac{\widehat{D}_\lambda(p_{\text{NOW}-1},p_{\text{NOW}})}{2}$
$\frac{\widehat{D}_\lambda(p_{\text{NOW}-1},p_{\text{NOW}})+\widehat{D}_\lambda(p_{\text{NOW}-1},p_{\text{NOW}})}{2}$

Figure 7: This figure shows how the pseudo-distance $\widehat{D}_\lambda$ between $p_{\text{NOW}-2}$ and $p_{\text{NOW}-1}$, and between $p_{\text{NOW}-1}$ and p_{NOW} changes after agent i has optimally placed $p_{\text{NOW}-1}$.

Then the A_i are the dynamics matrices for the system

$$\widehat{\mathbf{D}}_\lambda(t_2) = A_i \widehat{\mathbf{D}}_\lambda(t_1),$$

where $t_2 > t_1$ is the time when the interpolation point i is moved by a sensing agent to its new Voronoi center, assuming that between t_1 and t_2 no other interpolation point is moved. If at the same instant more interpolation points are relocated, then the dynamics matrix is the product of all the A_i that correspond to the relocated interpolation points. We can now relax the assumptions and we can consider slowly time-varying ∂Q. Let t_i^k be the k-th time that p_i is optimally placed by an agent. Before optimally placing p_i, the agent will project p_{i+1}. Because the boundary has changed, the pseudo-distance $\widehat{D}_\lambda(p_i, p_{i+1}, t_i^k)$ will differ from $\widehat{D}_\lambda(p_i, p_{i+1}, t_{i+1}^{k-1})$ by some noise $g(t_i^k - t_{i+1}^{k-1})$. Therefore the system is evolving according to a dynamical system of the form:

$$\widehat{\mathbf{D}}_\lambda(t_i^k)^+ = A_i \left(\widehat{\mathbf{D}}_\lambda(t_i^k) + \mathbf{e}_{i+1} g(t_i^k - t_{i+1}^{k-1}) \right), \tag{22}$$

where $\mathbf{e}_i$ is the column vector with null entries but the i-th component that is equal to 1. Let $\Delta T = \sup_{t \in \overline{\mathbb{R}}_+} \frac{L(\partial Q(t))}{v_{\min}}$. Note that $\Delta T < +\infty$ since by assumption the length of the boundary $\partial Q(t)$ is uniformly upperbounded. This means that at most after ΔT any interpolation point is updated at least once. Any time that an agent updates any interpolation point p_i the vector $\widehat{\mathbf{D}}_\lambda$ evolves according to (22), where $t_i^k - t_{i+1}^{k-1}$ is upperbounded by ΔT.

Because ΔT is finite, there exists a sequence of instants τ_k, with $k \in \mathbb{N}_0$, such that across the interval $[\tau_{k-1}, \tau_k]$ every interpolation point has been updated at least once by an agent. In other words:

$$\widehat{\mathbf{D}}_\lambda(k+1) = \mathcal{A}(k) \widehat{\mathbf{D}}_\lambda(k) + \mathbf{u}(k), \quad k \in \mathbb{N}_0, \tag{23}$$

where we identify $\tau_0 \equiv 0$, and $\tau_k \equiv k$, and where $\mathcal{A}(k) = \Pi_{j=1}^{M(k)} A_{j_k}$, $j_k \in \{1, \dots, n_{\text{ip}}\}$. The graph associated with $\mathcal{A}(k)$ is connected and $\mathcal{A}(k)$ is ergodic. Furthermore, $\mathcal{A}(k)$ is doubly stochastic because the product of double stochastic matrices. The value of the index j_k depends on the order in which the interpolation points are updated. It is easy to see that $n_{\text{ip}} \leq M(k) \leq n_{\text{a}} n_{\text{ip}}$. Consider now the disagreement vector $\mathbf{d}(k)$ defined in (20). Recalling (23), and that $\mathcal{A}(k)$ is doubly stochastic, we can write the update law of the disagreement $\mathbf{d}(k)$:

$$\mathbf{d}(k+1) = \mathcal{A}(k) \mathbf{d}(k) + \delta \mathbf{u}(k), \quad k \in \mathbb{N}_0, \tag{24}$$

where $\delta\mathbf{u}(k) = \mathbf{u}(k) - \frac{\mathbf{1}^T\mathbf{u}(k)}{n_{\rm ip}}\mathbf{1}$. Because $\mathcal{A}(k)$ is ergodic, the equilibrium point of the unperturbed system is the origin which means that, if $\mathbf{u}(k)$ is equal to $\mathbf{0}$, asymptotically the system (23) will reach $\frac{\mathbf{1}^T\widehat{\mathbf{D}}_\lambda(k)}{n_{\rm ip}}\mathbf{1}$.

Let $V(\mathbf{d}(k)) = \mathbf{d}(k)^T\mathbf{d}(k)$ be a Lyapunov function candidate for the dynamics of $\mathbf{d}(k)$. Then:

$$V(\mathbf{d}(k+1)) - V(\mathbf{d}(k)) = -\mathbf{d}(k)^T R(k)\mathbf{d}(k) + \delta\mathbf{u}(k)^T\delta\mathbf{u}(k) + 2\delta\mathbf{u}(k)^T\mathcal{A}(k)\mathbf{d}(k),$$

where $-R(k) = \left(\mathcal{A}(k)^T\mathcal{A}(k) - I_{n_{\rm ip}}\right)$. By Theorem 1 it can be proved that $R(k)$ is positive semidefinite and the only eigenvalue at the origin is associated with the eigenvector $\mathbf{1}$ that is orthogonal to $\mathbf{d}(k)$.

Let $\mathcal{A}_q$ be a generic element of the set of cardinality $n_{\rm ip}! + (n_{\rm ip}+1)! + \cdots + (n_{\rm ip}n_{\rm a})!$ containing all the possible matrices obtained by multiplying M matrices A_i, with $n_{\rm ip} \leq M \leq n_{\rm a}n_{\rm ip}$ and $i \in \{1,\dots,n_{\rm ip}\}$, such that the graph associated with $\mathcal{A}_q$ is connected. Let $r_q^j = \{r \in \mathbb{R}|\det\Big(rI_{n_{\rm ip}} - (\mathcal{A}_q^T\mathcal{A}_q - I_{n_{\rm ip}})\Big) = 0\} \setminus \{0\}$ and $r_q = \min_j |r_q^j|$, then $\exists\, \overline{r} = \min_q r_q < 0$ because we are considering a finite set of matrices. We can then upperbound $V(\mathbf{d}(k+1)) - V(\mathbf{d}(k))$ as follows:

$$V(\mathbf{d}(k+1)) - V(\mathbf{d}(k)) \leq -\alpha_3(\|\mathbf{d}(k)\|) + \sigma(\|\delta\mathbf{u}\|),$$

where $\alpha_3(\|\mathbf{d}(k)\|) = \frac{1}{2}\overline{r}\|\mathbf{d}(k)\|^2$, $\sigma(\|\delta\mathbf{u}\|) = (\frac{2}{\overline{r}}+1)\|\delta\mathbf{u}\|^2$. By [16] the system described by (24) is input-to-state stable. Since the system (24) is ISS, we can now relax the assumption that no error effect the calculation of $\widehat{C}_{\rm NOW-1}$ and still be able to conclude that $\widehat{\mathbf{D}}_\lambda$ will asymptotically get close to the equilibrium of the unforced system, i.e., the interpolation points are uniformly distributed according to $\widehat{D}_\lambda$. This is simply because also this error enters linearly in the system (24).

We can now prove the ISS property for the dynamics of δL.

Let us suppose that the $\partial Q(t)$ is time-invariant and that the sensing agents can actually compute without error the arc length distance between them and their clockwise and counterclockwise neighbors. The dynamics for $\mathbf{L}(t)$ can be derived as follows. Consider

$$\dot{L}(P_i(t), P_{i+1}(t)) = v_{i+1} - v_i, \tag{25}$$

where

$$v_{i+1} = v_0 + k\left(L\left(P_{i+1}, P_{i+2}\right) - L\left(P_i, P_{i+1}\right)\right), \tag{26}$$

$$v_i = v_0 + k\left(L\left(P_i, P_{i+1}\right) - L\left(P_{i-1}, P_i\right)\right). \tag{27}$$

Substituting now equations (26) and (27) in equation (25), we have:

$$\dot{L}(P_i(t), P_{i+1}(t)) = k\Big(L\left(P_{i+1}, P_{i+2}\right) - 2L\left(P_i, P_{i+1}\right) + L\left(P_{i-1}, P_i\right)\Big).$$

If the saturation on the speeds is not active, we have:

$$\dot{\mathbf{L}}(t) = k\begin{bmatrix} -2 & 1 & 0 & \dots & 1 \\ 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 1 & 0 & \dots & 1 & -2 \end{bmatrix}\mathbf{L}(t) = kA_L\mathbf{L}(t).$$

If we introduce the saturation on the speeds v_i, then the dynamics of $\mathbf{L}$ becomes:

$$\dot{\mathbf{L}}(t) = kA(c_1,\dots,c_{n_{\rm a}})\mathbf{L}(t),$$

where $A(c_1,\dots,c_{n_{\rm a}})$ is defined in Section 2, with $n = n_{\rm a}$, $\alpha = \frac{\min\{v_{\max}-v_0, v_0-v_{\min}\}}{kL(\partial Q)}$. To calculate the lowerbound of c_i, α, suppose that the saturation on the velocity is active for the sensing agent i. This is equivalent to saying that

$$v_i = v_0 + k_i'\left(L(P_i, P_{i+1}) - L(P_{i-1}, P_i)\right),$$

where $k'_i = kc_i$, $c_i \leq 1$. We can think of the saturation function as a change in the gain in the control law. If $v_i = v_{\max}$, then

$$k'_i = \frac{v_{\max} - v_0}{(L(P_i, P_{i+1}) - L(P_{i-1}, P_i))} \geq \frac{v_{\max} - v_0}{L(\partial Q)} \quad \Longrightarrow \quad c_i = \frac{k'_i}{k} \geq \frac{1}{k}\frac{v_{\max} - v_0}{L(\partial Q)} > 0.$$

If $v_i = v_{\min}$, then

$$k'_i = \frac{v_0 - v_{\min}}{(L(P_i, P_{i+1}) - L(P_{i-1}, P_i))} \geq \frac{v_0 - v_{\min}}{L(\partial Q)} \quad \Longrightarrow \quad c_i = \frac{k'_i}{k} \geq \frac{1}{k}\frac{v_0 - v_{\min}}{L(\partial Q)} > 0.$$

Therefore a lowerbound for the c_i is given by the $\min\{v_{\max} - v_0, v_0 - v_{\min}\}\frac{1}{kL(\partial Q)}$. It can be proved (see Lemma 2) that the new matrices $A(c_1, \ldots, c_{n_{\mathrm{a}}})$, like A_L, are negative semidefinite. The only eigenvalue at the origin is associated with the eigenvector $\mathbf{1}$.

Let us consider the disagreement $\delta\mathbf{L}$ as described by (21), then

$$\delta\dot{\mathbf{L}}(t) = A(c_1, \ldots, c_{n_{\mathrm{a}}})\delta\mathbf{L}(t), \tag{28}$$

and the candidate Lyapunov function $V(\delta\mathbf{L}(t)) = \delta\mathbf{L}^T(t)\delta\mathbf{L}(t)$, then

$$\dot{V}(\delta\mathbf{L}(t)) = 2\delta\mathbf{L}(t)A(c_1, \ldots, c_{n_{\mathrm{a}}})\delta\mathbf{L}(t) \leq 0,$$

where the equality holds only if the entries of $\delta\mathbf{L}(t)$ are all zero. Since c_i belong to a compact set, the matrices $A(c_1, \ldots, c_{n_{\mathrm{a}}})$ belong to a compact set, and since the eigenvalues of a matrix are a continuous function of its entries (see [17]) then there exists an upperbound $-\rho < 0$ for the eigenvalues that are different from zero and as a consequence:

$$\dot{V}(\delta\mathbf{L}(t)) \leq -\rho\|\delta\mathbf{L}(t)\|^2.$$

We can then conclude that the system (28) is exponentially stable.

Let us now analyze how the pursuit objective can still be achieved when the boundary is slowly-varying and when the instead of $L(P_i, P_{i+1})$ the agents use only the approximation $\widehat{L}(P_i, P_{i+1})$. Let $\mathbf{u}_d(t)$ be the vector that expresses the change in the arc length distance between any two consecutive agents due to the deformation of $\partial Q(t)$, and let $\mathbf{u}_d^{av}(t) = \frac{\mathbf{1}^T\mathbf{u}_d(t)}{n_{\mathrm{a}}}\mathbf{1}$, then

$$\mathbf{u}_d(t) = \delta\mathbf{u}_d(t) + \mathbf{u}_d^{av}(t).$$

Let $\mathbf{L}^{av}(t) = \frac{\mathbf{1}^T\mathbf{L}(t)}{n_{\mathrm{a}}}\mathbf{1}$, then:

$$\mathbf{L}(t) = \mathbf{L}^{av}(t) + \delta\mathbf{L}(t) = \mathbf{L}^{av}(0) + \mathbf{u}_d^{av}(t) + \delta\mathbf{L}(t),$$

and taking the derivative respect to time of both sides we have:

$$\dot{\mathbf{L}}(t) = \dot{\mathbf{u}}_d^{av}(t) + \delta\dot{\mathbf{L}}(t).$$

The quantity $\dot{\mathbf{u}}_d^{av}(t)$ is bounded because the arc length is a composition of smooth function in t and the parametrization of the boundary ∂Q. In the case of slowly time-varying boundary the variation in time of the vector $\mathbf{L}(t)$ is due, not only to the fact that the agents speed up and slow down as imposed by the algorithm, but also to the deformation of ∂Q:

$$\dot{\mathbf{L}}(t) = A(t)\widehat{\mathbf{L}}(t) + \dot{\mathbf{u}}_d(t) = A(t)(\mathbf{L}(t) + \mathbf{u}_i(t)) + \dot{\mathbf{u}}_d(t),$$

where $A(t) = A(c_1, \ldots, c_{n_{\mathrm{a}}})$, while $\mathbf{u}_i(t) \in \mathbb{R}^{n_{\mathrm{a}} \times 1}$ is the noise due to the fact that the agents do not know exactly the arc length distance between them and their neighbors, $\mathbf{L}$, but just an estimate through the interpolation points, $\widehat{\mathbf{L}}$. Using the change of variables in equation (21), and recalling that $A(t)\mathbf{1} = \mathbf{0}$, for all t, we have:

$$\dot{\mathbf{u}}_d^{av}(t) + \delta\dot{\mathbf{L}}(t) = A(t)\delta\mathbf{L}(t) + A(t)\delta\mathbf{u}_i(t) + \dot{\mathbf{u}}_d(t),$$

where $\delta\mathbf{u}_i = \mathbf{u}_i - \frac{\mathbf{1}^T\mathbf{u}_i(t)}{n_\mathrm{a}}\mathbf{1}$. It follows that

$$\delta\dot{\mathbf{L}}(t) = A(t)\delta\mathbf{L}(t) + A(t)\delta\mathbf{u}_i(t) + \delta\dot{\mathbf{u}}_d(t), \tag{29}$$

with $\delta\dot{\mathbf{u}}_d(t) = \dot{\mathbf{u}}_d(t) - \dot{\mathbf{u}}_d^{av}(t)$. The system described by equation (29) is input-to-state stable (with input $A(t)\delta\mathbf{u}_i(t) + \delta\dot{\mathbf{u}}(t)_d$) because (i) the unforced system is exponentially stable, and (ii) the right-hand-side of (29) is differentiable and uniformly globally Lipschitz in $\delta\mathbf{L}$ and $A(t)\delta\mathbf{u}_i(t) + \delta\dot{\mathbf{u}}(t)_d$, (see [18]).

The ISS property guarantees that if $\mathbf{u}_i$, the error between $\mathbf{L}$ and its estimate $\widehat{\mathbf{L}}$, is small and if $\mathbf{u}_d$ is smaller, then the agents will get close to the equilibrium point of (28), which corresponds to have the agents uniformly distributed according to the arc length. The larger n_ip is and the slower the deformation of ∂Q is, then the smaller $\mathbf{u}_i$ and $\mathbf{u}_d$ are, and the closer to $\mathbf{0}$ the disagreement $\delta\mathbf{L}$ will get asymptotically. $\square$

4.3 Simulations

In this section we present results of two different simulations obtained with the implementation of the Estimate Update and Pursuit Algorithm. In the first simulation the boundary ∂Q is time invariant, while in the second is time varying.

4.3.1 Time-invariant boundary

In this simulation we use $n_\mathrm{a} = 3$ sensing agents to have an approximation of the non-convex boundary ∂Q described by (18). The outcome is shown in Figure 8. In order to calculate their speeds, the sensing agents use $v_0 = 1$, and $k = 0.05$. The saturation function for the speed has lower limit $v_\mathrm{min} = 0.5$ and upper limit $v_\mathrm{max} = 2$. The number of interpolation points is $n_\mathrm{ip} = 30$, while $\lambda = \frac{10}{11}$. The simulation time is 50 seconds and the sampling time 0.01 seconds. The plots in Figure 8 corresponds to the positions of the interpolation points and the sensing agents at four different instants: $t = 0$, $t = 20$, $t = 40$, and $t = 50$ seconds. The interpolation points p_{NOW^i} for $i \in \{1, \dots, n_\mathrm{a}\}$ coincide with the positions of the sensing agents. The other interpolation points are randomly distributed on the boundary. In the last frame one can also see the approximating polygon and how close it is to the actual boundary.

Since the pseudo-distance D_λ and the arc length L can be calculated after the simulation is completed, we use D_λ and L instead of their estimate $\widehat{D}_\lambda$ and $\widehat{L}$ to show the algorithm performance. Figure 9 does indeed show the convergence of the algorithm. In the first plot we can see that the consensus on the pseudo-distance $D_\lambda(p_i, p_{i+1})$, between any two consecutive interpolation points, is reached. The quantity $\max_{i\in\{1,\dots,n_\mathrm{ip}\}} D_\lambda(p_i, p_{i+1}) - \min_{i\in\{1,\dots,n_\mathrm{ip}\}} D_\lambda(p_i, p_{i+1})$ does not vanish because of numerical errors in the estimate $\widehat{D}_\lambda$. The second plot shows how the agents get uniformly spaced along the boundary. The steady state values of the arc length distances oscillates around 8.3 which is the target value. The noise is again due to the fact that the agents only estimate the arc length using the positions of the interpolation points.

4.3.2 Slowly time-varying boundary

In this simulation we used $n_\mathrm{a} = 3$ sensing agents to have an approximation of the non-convex boundary $\partial Q(t)$ described by:

$$\gamma(\theta, t) = \left(2\left(1 - \frac{t}{t_f}\right) + \Big(2 + \cos(10\pi\theta) + 0.5\sin(4\pi\theta)\Big)\frac{t}{t_f}\right)\begin{bmatrix}\cos(2\pi\theta)\\ \sin(2\pi\theta)\end{bmatrix},$$

with $\theta \in [0, 1)$, $t_f = 200$ seconds as shown in Figure 10. The values of v_0, k, v_min, v_max and λ are respectively: 1, 0.05, 0.5, 2, and $\frac{10}{11}$. The simulation time is 200 seconds, the sampling time 0.01 seconds. The plots in

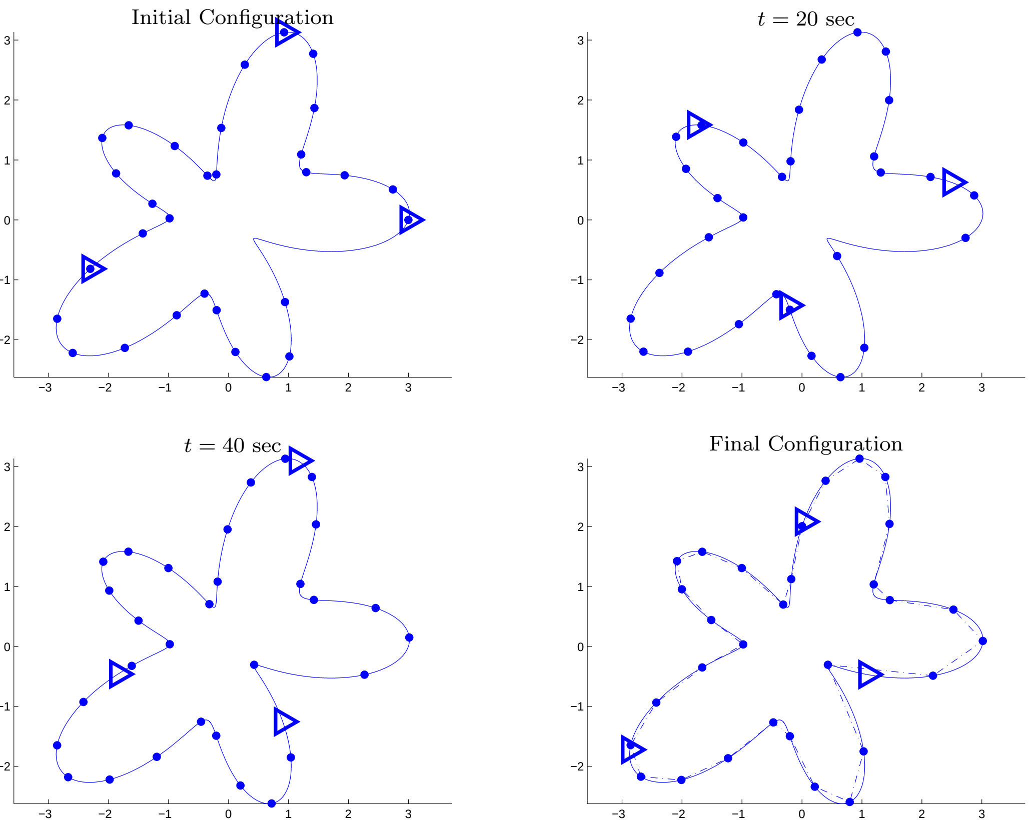


Figure 8: This figure shows four different instants of 50 seconds simulation obtained by the implementation of the Estimate Update and Pursuit Algorithm with $n_{\mathrm{a}} = 3$, $n_{\mathrm{ip}} = 30$, $v_0 = 1$, $k = 0.05$, $\lambda = \frac{10}{11}$. ∂Q is time invariant. The sensing agents position is represented by the triangles and are initialized to be on the boundary ∂Q. In the last frame also the approximating polygon is shown.

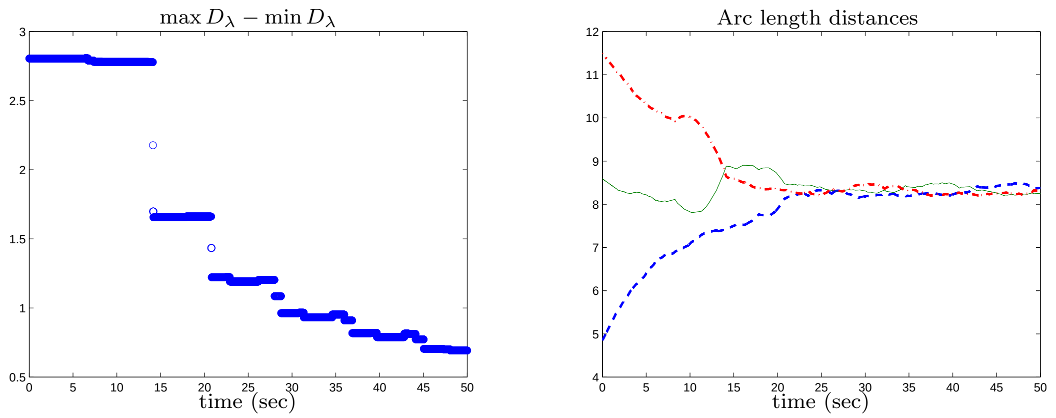


Figure 9: Estimate Update and Pursuit Algorithm This plots refers to the case of ∂Q being time-invariant. In the first plot from right it is shown the error $\max_{i \in \{1,\dots,n_{\mathrm{ip}}\}} D_\lambda(p_i, p_{i+1}) - \min_{i \in \{1,\dots,n_{\mathrm{ip}}\}} D_\lambda(p_i, p_{i+1})$ vs time. In the second plot we show the arc length distances between the three sensing agents.

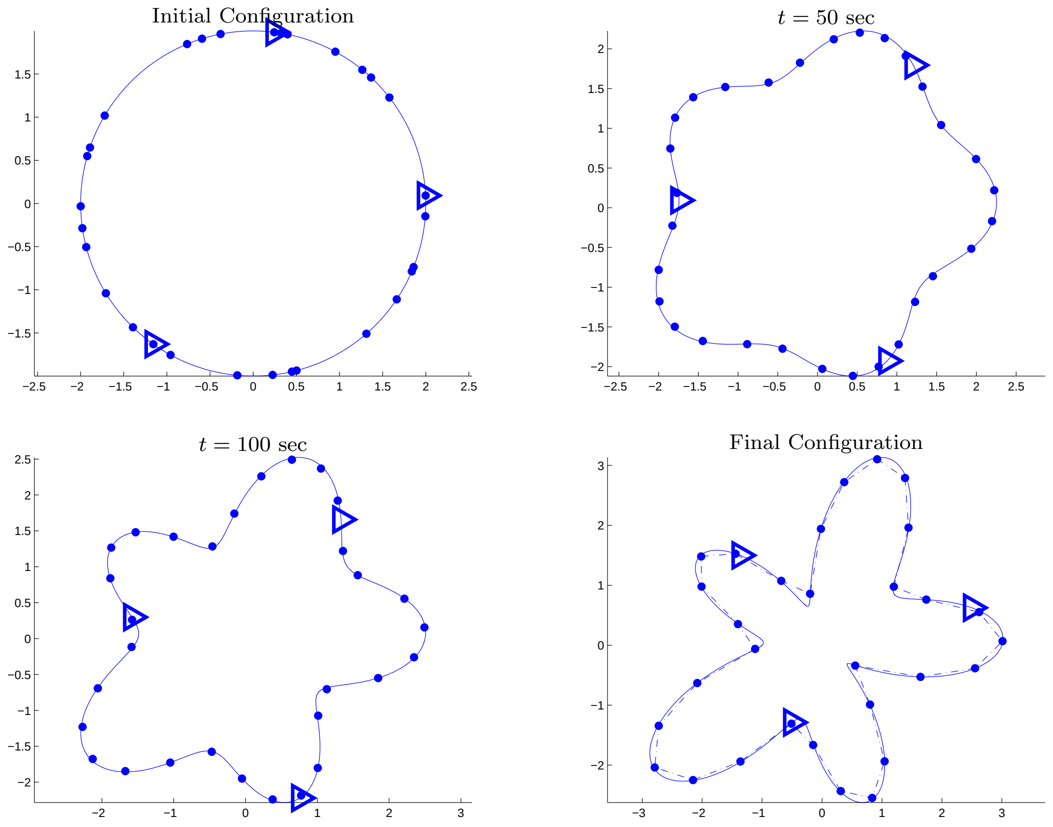


Figure 10: This figure shows four different instants of the 200 seconds simulation obtained by implementing the Estimate Update and Pursuit Algorithm with $n_{\mathrm{a}} = 3$, $n_{\mathrm{ip}} = 30$, $v_0 = 1$, $k = 0.05$, $\lambda = \frac{10}{11}$. The boundary ∂Q is slowly time-varying in this case. The sensing agents positions are represented by triangles and initialized to be on the boundary ∂Q. The last frame also shows the approximating polygon.

Figure 10 correspond to the positions of the interpolation points and the sensing agents at four different instants, $t = 0$, $t = 50$, $t = 100$, and $t = 200$ seconds respectively. The algorithm is initialized with the agents on the boundary. The interpolation points p_{NOW^i} coincide with the positions of the sensing agents. The other interpolation points are randomly distributed. In the last frame we can also see the approximating polygon and how close to the actual boundary is. From the frames in Figure 10 it is clear that the sensing agents can adapt as ∂Q changes.

The pseudo-distance D_λ is well defined only if the interpolation points belong to the boundary ∂Q. Since the boundary changes with time, the interpolation points are only for some time on the boundary after a sensing agents has projected them. So, we consider as pseudo-distance between any two consecutive interpolation points in a certain time τ the pseudo-distance between their radial projection onto $\partial Q(\tau)$. The disagreement in the placement of the interpolation points, where D_λ is redefined as just explained, is shown in the first plot of Figure 11.

The arc length between any two consecutive sensing agents is shown in the second plot of Figure 11. The three distances increase with time because $L(\partial Q)$, the total length of the boundary, increases with time.

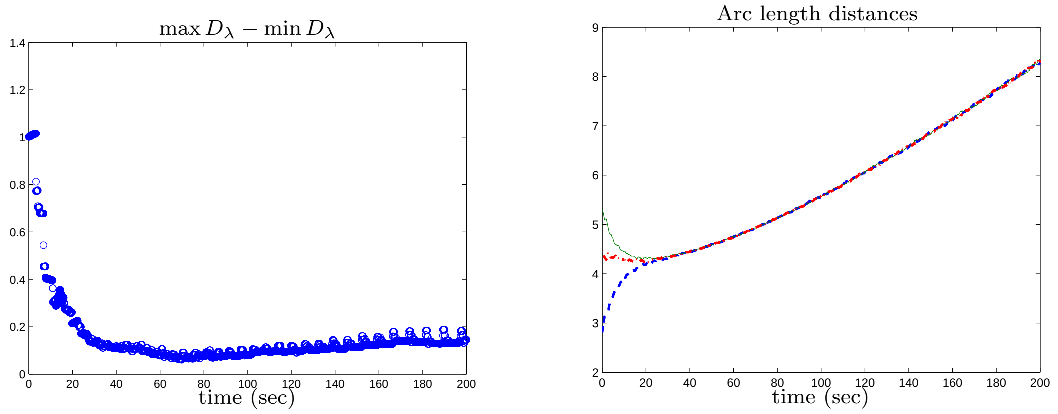


Figure 11: Estimate Update and Pursuit Algorithm. This figure refers to the case of ∂Q being slowly time-varying. In the first plot from the right we shown the error $\max_{i\in\{1,\dots,n_{\mathrm{ip}}\}} D_\lambda(p_i,p_{i+1}) - \min_{i\in\{1,\dots,n_{\mathrm{ip}}\}} D_\lambda(p_i,p_{i+1})$ vs time. The second plot shows the arc length distances between the three sensing agents.

5 Conclusions

In this paper we have addressed the problem of boundary estimation and tracking by means of robotic sensors. We have presented algorithms to position the agents or interpolation points along the boundary in such a way as to obtain an approximating polygon with some optimality features. It is proven that the best approximating polygon for convex bodies, according to the symmetric difference δ^S, is the polygon whose vertices are uniformly distributed according to the metric $\int \rho^{2/3}(\theta)d\theta$. We used a new metric $D_\lambda = \int(\lambda\kappa^{1/3}d\ell + (1-\lambda)d\ell)$ that differs from the other one in two aspects. Motivated by the fact that $\int_{\partial Q}\kappa^{1/3}d\ell = \int_0^{2\pi}\rho^{2/3}(\theta)d\theta$ for convex bodies, we substitute ρ with κ because ρ is not well defined for a non-convex bodies. We further modified the metric introducing the arc length so that for any two non-coincident points the pseudo-distance D_λ is always different from zero. In our first ideal scenario the agents know the boundary. The corresponding algorithm leads the robots to positions that are uniformly spaced according to the pseudo-distance D_λ. Using tools from network consensus analysis, the algorithm is proven to converge for large communication radiuses; simulations illustrate that the algorithm converges also when the communication range is limited.

In the second and more realistic scenario, the mobile agents are equipped with sensors that provide local information on the tangent and curvature of the boundary. The second algorithm allows the robots to place a set of interpolation points uniformly spaced according to the estimate of the pseudo-distance D_λ. The position of the interpolation points is stored in a data fusion center and is available on-demand to the agents. The vertices of the approximating polygon are the interpolation point positions. The algorithm is proven to converge even if the boundary is slowly-moving. As in the first scenario, tools from consensus analysis allow us to prove the correctness of the second algorithm.

NEW PARAGRAPH

The existence of a central data fusion center is not a critical ingredient in the design of the algorithm. Indeed, one can envision the following equivalent scenario: the agents communicate the updated interpolation points to their clockwise neighboor, instead of exchanging them with the data fusion center. In such a distributed setting, a stationary user could reconstruct the approximating polygon by communicating to all the agents as they pass by a fixed spatial location. Future research will explore this idea more in detail. $\star$

Acknowledgments

This material is based upon work supported in part by ONR YIP Award N00014-03-1-0512 and NSF SENSORS Award IIS-0330008. Sonia Martínez's work was supported in part by a Fulbright Postdoctoral Fellowship from the Spanish Ministry of Education and Science.

References

[1] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Proc. of the Conference on Cooperative Control and Optimization*, (Gainesville, FL), Dec. 2002.

[2] A. L. Bertozzi, M. Kemp, and D. Marthaler, "Determining environmental boundaries: asynchronous communication and physical scales," in *Proceedings of the 2003 Block Island Workshop on Cooperative Control* (V. Kumar, N. E. Leonard, and A. S. Morse, eds.), vol. 309 of *Lecture Notes in Control and Information Sciences*, pp. 25–42, New York: Springer Verlag, 2004.

[3] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *American Control Conference*, (Portland, OR), pp. 3500–3505, June 2005.

[4] D. W. Casbeer, S.-M. Li, R. W. Beard, R. K. Mehra, and T. W. McLain, "Forest fire monitoring with multiple small UAVs," in *American Control Conference*, (Portland, OR), pp. 3530–3535, June 2005.

[5] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. Mclain, S.-M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Sciences*, 2005. To appear.

[6] F. Zhang and N. E. Leonard, "Generating contour plots using multiple sensor platforms," in *IEEE Swarm Intelligence Symposium*, (Pasadena, CA), pp. 309–316, June 2005.

[7] P. M. Gruber, "Approximation of convex bodies," in *Convexity and its Applications* (P. M. Gruber and J. M. Willis, eds.), pp. 131–162, Birkhäuser Verlag, 1983.

[8] P. M. Gruber, "Aspect of approximation of convex bodies," in *Handbook of Convex Geometry* (P. M. Gruber and J. M. Willis, eds.), vol. A, pp. 319–345, Oxford, UK: Elsevier, 1993.

[9] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[10] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *IEEE Conf. on Decision and Control*, (Seville, Spain), pp. 2996–3000, Dec. 2005.

[11] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[12] D. E. McLure and R. A. Vitale, "Polygonal approximation of plane convex bodies," *Journal of Mathematical Analysis and Applications*, vol. 51, no. 2, pp. 326–358, 1975.

[13] L. Chen and J. Xu, "An optimal streamline diffusion finite element method for a singularly perturbed problem," in *AMS Contemporary Mathematics Series: Recent Advances in Adaptive Computation*, (Hangzhou, China), 2004.

[14] L. Chen, "Optimal interpolation error estimates and applications." Personal Communication, Jan. 2005.

[15] M. Fielder, *Special Matrices and their Applications in Numerical Mathematics*. Martinus Nijhoff Publishers, 1986.

[16] Z.-P. Jiang and Y. Wang, “Input-to-state stability for discrete-time nonlinear systems,” *Automatica*, vol. 37, pp. 857–869, 2001.

[17] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge, UK: Cambridge University Press, 1985.

[18] H. K. Khalil, *Nonlinear Systems.* Englewood Cliffs, NJ: Prentice Hall, 2 ed., 1995.

Name:	Estimate Update and Pursuit Algorithm
Goal:	Uniformly distribute the interpolation points according to the pseudo-distance $\widehat{D}_\lambda$, and the sensing agents according to the arc length $\widehat{L}$.
Data:	Location of the interpolation points, unitary tangent vector at ∂Q at those points, last value of $\widehat{D}_\lambda$ between any two consecutive interpolation points, local tangent and local curvature of the boundary ∂Q.
Requires:	At $t_0 = 0$ p_i lie on ∂Q and $\widehat{D}_\lambda$ between any two interpolation points is known.

Assume data is as stated in (19). At every sensing instant, the agent at position $P_i(t) = P(t)$ performs:

```
1:  if $\widehat{D}_\lambda(o_{L+1}, P(t)) > 2\widehat{D}^-_\lambda(p_{\textsc{now}-1}, p_{\textsc{now}})$, then
2:      update the set of observations $\textsc{nextbuffer}^+ := \textsc{nextbuffer} \cup \{P(t)\}$,
3:  else
4:      update the set of observations $\textsc{bufferarc}^+ := \textsc{bufferarc} \cup \{P(t)\}$.
5:  end if
6:  estimate $\widehat{\gamma'}(P(t))$, $\widehat{\kappa}(P(t))$, and $\widehat{D}_\lambda(o_{L+M+T}, P(t))$.
7:  if $\textsc{nextbuffer} \neq \emptyset$ and $p_{\textsc{now}^i} \neq p_{\textsc{now}^{i+1}-2}$ then
8:      update the interpolation point $p_{\textsc{now}}$ by projecting it onto $\partial Q$:
            $$p^+_{\textsc{now}} := o_{\bar{j}}, \quad o_{\bar{j}} = \text{argmin}_{o_j \in \textsc{bufferarc}} \|(o_j - p_{\textsc{now}}) \cdot \mathbf{t}^-_{\textsc{now}}\|,$$
9:      update the set BUFFERARC and generate the set NOWARC by:
            $$\begin{aligned} \textsc{bufferarc}^+ &:= \textsc{bufferarc} \setminus \{o_{L+1}, \dots, o_{\bar{j}}\}, \\ \textsc{nowarc}^+ &:= \{o_{L+1}, \dots, o_{\bar{j}}\}, \end{aligned}$$
10:     calculate $\widehat{C}_{\textsc{now}-1} := o_{\bar{k}}$ and update $p_{\textsc{now}-1}$ by $p^+_{\textsc{now}-1} := o_{\bar{k}}$,
11:     communicate with data center: transmit $p_{\textsc{now}-1}$, $p_{\textsc{now}}$, $\widehat{\gamma'}(p_{\textsc{now}-1})$, $\widehat{D}_\lambda(p_{\textsc{now}-2}, p_{\textsc{now}-1})$,
        $\widehat{D}_\lambda(p_{\textsc{now}-1}, p_{\textsc{now}})$ and receive $p_{\textsc{now}+1}$, $\widehat{\gamma'}(p_{\textsc{now}+1})$, $\widehat{D}_\lambda(p_{\textsc{now}}, p_{\textsc{now}+1})$,
12:     update the counter NOW and the set LASTARC by
            $$\textsc{now}^+ := \textsc{now} + 1, \ \textsc{lastarc}^+ := \{o_{\bar{k}}, \dots, o_{\bar{j}}\},$$
13:     update the sets BUFFERARC and NEXTBUFFER as follows:
14:     if $\exists o_q \in \textsc{bufferarc} \cup \textsc{nextbuffer}$ s.t. $\widehat{D}^-_\lambda(o_{\bar{j}+1}, o_q) \geq 2\widehat{D}_\lambda(p_{\textsc{now}^+-1}, p_{\textsc{now}^+})$, then
15:         $\textsc{bufferarc}^+ := \{o_{\bar{j}+1}, \dots, o_q\}$, $\textsc{nextbuffer}^+ := \{o_{q+1}, \dots, o_{L+M+T}\}$,
16:     else
17:         $\textsc{bufferarc}^+ := \textsc{bufferarc} \cup \textsc{nextbuffer}$, $\textsc{nextbuffer}^+ := \emptyset$.
18:     end if
19: end if
20: communicate with $P_{i+1}$ and $P_{i-1}$: receive $\textsc{now}^{i+1}$, $\textsc{now}^{i-1}$, transmit $\textsc{now}^i$. Communicate with the
    data center: receive the positions of the interpolation points with id between $\textsc{now}^{i-1}$ and $\textsc{now}^{i+1}$.
21: calculate $v_i(t)$: $v_i(t) = \text{sat}(v_0 + k(\widehat{L}(P_i, P_{i+1}) - \widehat{L}(P_{i-1}, P_i)))$.
```